UNIVERSITÀ DEGLI STUDI DI PISA

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

UNO STRUMENTO PER L'ANALISI DELLE DIPENDENZE E LA VERIFICA DEL FLUSSO DI INFORMAZIONE SICURO NEL CIL

Primo Relatore: Prof.ssa C. Bernardeschi

Secondo Relatore: Prof. M. Avvenuti

Candidato: Luca Rapalini

ANNO ACCADEMICO 2003–2004

Ai miei genitori

Indice

Riassunto analitico **ix**

Prefazione **x**

Introduzione **xi**

I Microsoft .NET **1**

1 Il Framework .NET **2**

1.1 Che cos'è il Framework .NET? 2

1.2 CLR (Common Language Runtime) 3

1.3 La libreria di classi 4

1.4 Assembly, Tipi e Namespace 5

2 La sicurezza in Microsoft .NET **8**

2.1 Le basi per la sicurezza 8

2.2 Evidence Based Security 9

2.2.1 Evidence (Prove dell'assembly) 9

2.2.2 Policy (Politiche di sicurezza) 10

2.2.3 Permissions (Sistema di autorizzazioni) 10
2.3 Code Access Security 11
2.4 Role Based Security 11
2.5 Isolate Storage: Spazi di archiviazione isolati 12

II SIF: Flusso sicuro delle informazioni 13

3 Flusso sicuro delle informazioni 14
3.1 Introduzione 14
3.2 Il problema 14
3.3 Funzionamento del programma d'esempio 16
3.4 SIF: Security Information Flow 18

4 DepComputerIL: tool per il calcolo delle dipendenze 21
4.1 Introduzione 21
4.2 Set di istruzioni trattate 22
4.3 Concetti di base 22
4.3.1 Control-Flow Graph 24
4.3.2 Immediate Post Dominators (IPDs) 26
4.3.3 Dipendenze di un'istruzione 26
4.4 Funzionamento del tool 27
4.5 Dettagli architetturali e implementativi 29
4.5.1 ILParser.dll 29
Parser 31
ILMethodBody 32

ILInstruction 33
4.5.2 DepComputerIL 33
Node 33
CFG 35
DepComputer 36

5 Test svolti sul DepComputerIL 38
5.1 Due casi semplici 38
5.1.1 Ciclo 38
5.1.2 If-Else 43
5.2 Un caso più complesso 47

III Verifica del flusso di informazione sicuro: ESIF-DotNet 51

6 ESIFDotNet: tool per la tutela delle informazioni private 52
6.1 Introduzione 52
6.2 Il SIFDotNet 52
6.2.1 Struttura interna 54
6.3 Dal SIFDotNet al ESIFDotNet 55
6.4 L'architettura di alto livello 56
6.5 Dettagli architetturali e implementativi 57
6.5.1 Security Manager 57
6.5.2 Abstract Interpreter 61
AbstrInterpreter 61

MethodVerifier 62
DescError 64
6.6 Problema delle librerie di sistema 67

7 Applicazione dell'ESIFDotNet ad un caso reale 68
7.1 Il controllo gestito CodFisc.dll 68
7.2 Una prima analisi di CodFisc.dll 70
7.3 Analisi più selettiva di CodFisc.dll 75

8 Conclusioni 79

A Analisi del metodo *SendData (DatiUtente userdata)* 82
A.1 Livelli di sicurezza tutti *Low* 82
A.2 Livelli più restrittivi 106

Elenco delle figure

1.1 Lo stack delle tecnologie 2

1.2 Tutti i linguaggi del CLR compilano in IL 4

1.3 Solo in fase di esecuzione il codice IL viene compilato in codice nativo 5

1.4 Formato di un Assembly .NET 7

3.1 Programma per il calcolo del codice fiscale, implementato come controllo gestito 15

3.2 Infrastruttura necessaria al funzionamento del programma . . . 17

3.3 I dati, inviati dal client, vengono intercettati dal server remoto, che, per non destare troppo sospetto, ritorna effettivamente il codice fiscale 18

3.4 Un caso di flusso implicito in un set di istruzioni CIL 20

4.1 Istruzioni trattate 23

4.2 Grafo e possibili sequenze di esecuzione 24

4.3 Nodi virtuali START e END 25

4.4 Esempio di target con 4 archi entranti 25

4.5 Esempio di IPD e PD 26
4.6 Esempio di dipendenza 27
4.7 Architettura di alto livello 28
4.8 ParserIL: sono visibili i Tipi esportati 30
4.9 ParserIL: Sequenza delle azioni intraprese in seguito ad una richiesta di servizio 31
4.10 Struttura della classe Node 34
4.11 Struttura della classe CFG 35
4.12 Struttura della classe DepComputer 37

5.1 Control-flow graph di un ciclo 44
5.2 Control-flow graph di un'istruzione if-else 46
5.3 Grafo del caso complesso 50

6.1 Architettura del ESIFDotNet 57
6.2 Struttura del sottosistema Security manager 58
6.3 Struttura interna del Tipo abstrInterpeter 61
6.4 Struttura interna del Tipo MethodVerifier 63
6.5 Albero di ereditarietà tra le classi di descrizione d'errore 65

Riassunto analitico

In questa tesi viene affrontato il problema dell'analisi statica del codice per rilevare il trattamento improprio di informazioni personali da parte di applicazioni .NET.

Il flusso di propagazione delle informazioni viene tracciato in base ad impostazioni di livello di sicurezza, fornite in modo manuale o automatico ai Tipi definiti nell'applicazione .NET.

A partire dalle informazioni di sicurezza vengono definite una serie di regole atte a formalizzare il comportamento del codice .NET in questo nuovo contesto.

Obiettivo finale del lavoro è stato lo sviluppo di uno strumento in grado di rilevare possibili violazioni della riservatezza dei dati, dovute all'esecuzione di programmi che hanno accesso ad informazioni riservate.

Prefazione

Microsoft .NET è un insieme di tecnologie concepite per garantire l'utilizzo e l'interconnessione di sistemi software in totale sicurezza. Il meccanismo di "Code Access Security" e "Role Based Security", che garantiscono un sistema di protezione basato sulla tipologia e provenienza del codice, forniscono un sistema di protezione, che arricchisce le politiche di sicurezza del Sistema Operativo.

Tuttavia i precedenti meccanismi non controllano la propagazione delle informazioni a cui è stato concesso l'accesso e informazioni riservate possono essere memorizzate dall'applicazione, per esempio, in risorse di dominio pubblico.

Questo problema, noto in letteratura come "Secure Information Flow" [14], è stato affrontato per l'architettura Java in [2].

Questa tesi propone i risultati ottenuti dallo studio dell'architettura .NET ed espone le problematiche connesse con il flusso di informazione sicuro, di cui il codice .NET non è esente. Si descrive, infine, l'implementazione ed il funzionamento di due Tool: il primo per il calcolo delle dipendenze fra le isruzioni del codice "Common Intermediate Language" (CIL) contenuto negli Assembly .Net, il secondo per effettuare un'analisi statica del codice degli Assembly .Net, al fine di individuare possibili violazioni delle informazioni informazioni personali.

Introduzione

La grande diffusione della banda larga, la maggiore disponibilità di risorse di calcolo e la grande richiesta di applicazioni hanno modificato le attuali tendenze nel settore del software che si sono orientate alla promozione di applicazioni lato client in grado di offrire i più svariati servizi.

In questo contesto diventa di primaria importanza il controllo dell'affidabilità del codice dell'applicazione e delle risorse alle quali si potrebbe accedere.

Le architetture di protezione tradizionali si basano sugli account utente per il controllo degli accessi e la definizione del livello di isolamento dei sistemi. Entro questi limiti, al codice vengono comunque assegnati diritti di accesso completi sulla base del presupposto che tutto il codice eseguito da un particolare utente è da considerarsi ugualmente affidabile. Sfortunatamente, l'isolamento del codice in base agli utenti non è sufficiente per proteggere un programma da un altro, in particolare se entrambi i programmi vengono eseguiti nel contesto di sicurezza dello stesso utente.

Microsoft .NET è un insieme di tecnologie che trovano un supporto per lo sviluppo nel FrameWork .NET.

Le applicazioni sviluppate utilizzando il FrameWork .NET sono distribuite in un linguaggio intermedio orientato agli oggetti chiamato Common Inter-

mediate Language (CIL) o, anche, Microsoft Intermediate Language (MSIL) ed eseguite da un ambiente di esecuzione, praticamente una virtual machine, chiamato Common Language Runtime (CLR).

Microsoft .NET propone una soluzione al problema della sicurezza che trova un giusto equilibrio tra i tradizionali meccanismi di protezione.

Questi meccanismi, però, non controllano la propagazione delle informazioni a cui è stato concesso l'accesso: informazioni riservate possono essere memorizzate dall'applicazione, per esempio, in risorse di dominio pubblico.

In questa tesi viene analizzato il problema della tutela della riservatezza delle informazioni private, noto come flusso di informazione sicuro (SIF) in applicazioni .NET. Per individuare la presenza di flussi illeciti di informazioni viene utilizzato un approccio per la verifica della sicurezza delle informazioni sviluppato in [2] e rielaborato per tenere conto delle differenze introdotte dal nuovo ambiente. Si propongono, inoltre, due tool:

- *DepComputerIL*: capace di calcolare le dipendenze tra le istruzioni del Common Intermediate Language, così da facilitare l'individuazione di quello che in letteratura è noto come *flusso implicito*;
- *ESIFDotNet*: derivato da un tool precedentemente realizzato, il *SIFDotNet*, è in grado di eseguire un'analisi iterativa del Secure Information Flow nel Common Intermediate Language.

La tesi è così organizzata:

- Capitolo 1 - Il Framework .NET
- Capitolo 2 - La sicurezza in Microsoft .NET

- Capitolo 3 - Flusso sicuro delle informazioni
- Capitolo 4 - DepComputerIL: tool per il calcolo delle dipendenze
- Capitolo 5 - Test svolti sul DepComputerIL
- Capitolo 6 - ESIFDotNet: tool per la tutela delle informazioni private
- Capitolo 7 - Applicazione di ESIFDotNet ad un caso reale
- Capitolo 8 - Conclusioni

Parte I

Microsoft .NET

Capitolo 1

Il Framework .NET

1.1 Che cos'è il Framework .NET?

Il termine .NET Framework si riferisce al gruppo di tecnologie che costituiscono la base di sviluppo della piattaforma Microsoft .NET [4]. Le principali tecnologie di questo gruppo sono rappresentate dal run-time e dalla libreria di classi (BCL Base Class Library), come illustrato nella Figura 1.1.

Figura 1.1: Lo stack delle tecnologie

1.2 CLR (Common Language Runtime)

CLR è responsabile dei servizi di run-time, tra cui l'integrazione del linguaggio, l'applicazione della protezione e la gestione della memoria, dei processi e dei thread. Svolge, inoltre, un importante ruolo in fase di sviluppo in quanto grazie a particolari funzionalità, tra cui la gestione della security, l'assegnazione di nomi sicuri, la gestione delle eccezioni tra più linguaggi, l'associazione dinamica dei dati e altre funzioni, consente agli sviluppatori di scrivere una minore quantità di codice per convertire la logica business in un componente riutilizzabile.

Il run-time è responsabile della gestione del codice e di fornire servizi al codice stesso durante la sua esecuzione. I linguaggi di programmazione .NET, tra cui Visual Basic .NET, Microsoft Visual `C#`^{TM}, le estensioni gestite C++ e molti altri linguaggi di fornitori diversi, utilizzano i servizi e le funzionalità .NET mediante un set di classi unificate.

La base del funzionamento del riutilizzo binario del codice in .NET è costituita dal Common Language Runtime, un ambiente d'esecuzione comune per tutti i linguaggi.

Il codice eseguito dal runtime parte dalla compilazione del codice intermedio detto IL (Intermediate Language). Tutti i compilatori dei linguaggi del CLR compilano in IL e quindi dispongono di un ambiente d'esecuzione comune con tipi di dati comuni, Figura 1.2.

In fase di esecuzione il linguaggio intermedio viene compilato in codice nativo della piattaforma sulla quale .NET è in esecuzione, Figura 1.3, grazie ad un JITer (Just in Time Compiler). La compilazione avviene in fase di runtime, solo per le parti di codice eseguite e, generalmente, una sola volta nel caso in

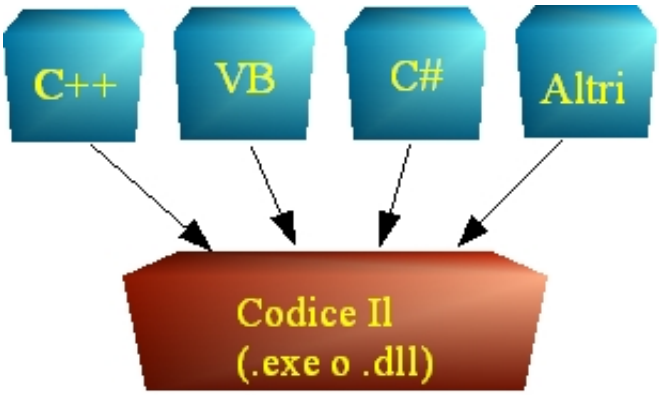

Figura 1.2: Tutti i linguaggi del CLR compilano in IL

cui la stessa parte di codice venga eseguita più volte nell'ambito della stessa istanza.

Questa tecnica di compilazione garantisce numerosi vantaggi, tra cui:

1. Il codine nativo viene compilato in base alla tipologia della macchina su cui viene eseguito: questo permette, ad esempio, un ottimizzazione del codice su macchine già datate

2. Vengono compilate solo le parti del codice effettivamente utilizzate

1.3 La libreria di classi

Le classi unificate .NET rappresentano la base su cui vengono create le applicazioni indipendentemente dal linguaggio utilizzato.

La librerie di classi è orientata agli oggetti e i suoi tipi, quindi, consentono di effettuare una gamma di attività comuni di programmazione, incluse operazioni quali la gestione di stringhe, la raccolta di dati, la connettività al database

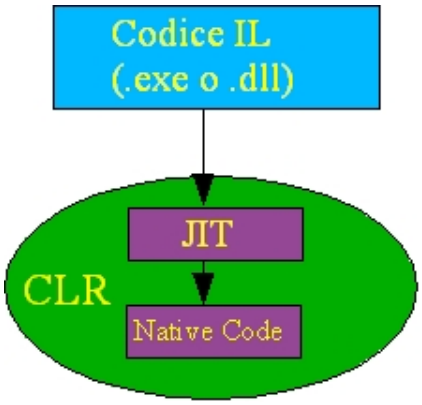

Figura 1.3: Solo in fase di esecuzione il codice IL viene compilato in codice nativo

e l'accesso a file. Oltre a queste attività comuni, la libreria di classi include tipi che supportano vari scenari di sviluppo specializzati. È possibile, ad esempio, utilizzare Framework .NET per sviluppare i seguenti tipi di applicazioni e servizi:

- applicazioni console
- applicazioni GUI Windows (Windows Form)
- applicazioni ASP.NET
- servizi Web XML
- servizi Windows

1.4 Assembly, Tipi e Namespace

Le unità binarie in .NET prendono il nome di Assembly, indipendentemente dal fatto che si tratti di .exe o .dll.

Alla base del contenuto degli assembly abbiamo i Tipi che possiamo utilizzare nella programmazione .NET (Strutture, Classi, Delegates, etc).

Un assembly può essere considerato come un insieme di Tipi e risorse unitariamente organizzati e descritti con uno specifico entry point che può essere eseguito o utilizzato per accedere ai tipi esportati (exe o dll).

I tipi all'interno di un Assembly sono organizzati in Namespace, che li raggruppano e ne semplificano l'organizzazione e l'utilizzo. Gli stessi tipi contenuti nelle classi di base di .NET sono organizzati in Namespace.

I Namespace permettono anche un organizzazione gerarchica. Ad esempio il Namespace principale dei servizi base è contenuto in System, i tipi per la costruzione delle finestre Windows sono contenuti nel Namespace WinForms che a sua volta è contenuto in System.

Una delle caratteristiche più importanti degli Assembly sta nella loro capacità di autodescriversi. Questa caratteristica è alla base della semplificazione della gestione del versioning e della distribuzione delle applicazioni. In ogni unità binaria .NET, infatti, esistono metadati che consentono la completa descrizione della struttura del componente e dei tipi in esso contenuti e i riferimenti a risorse e assembly che gli occorrono per poter funzionare (ovvero gli assembly da cui è dipendente).

Il file di un assembly è essenzialmente strutturato come rappresentato nella Figura 1.4.

Il manifesto contiene le informazioni di dipendenza da altri assembly e descrive caratteristiche e versione dei fornitori di servizi. Nel manifesto, poi, abbiamo le informazioni relative al componente vero e proprio, ovvero versione,

Figura 1.4: Formato di un Assembly .NET

originatore, etc.

La parte di Type Metadata contiene informazioni relative ai tipi contenuti nell'assembly. Questa parte può essere estesa con l'inserimento di attributi personalizzati.

Un assembly può essere costituito anche da più file binari, in questo caso l'architettura conterrà sempre un solo manifesto e sarà distribuito su più elementi binari di tipi e risorse.

Capitolo 2

La sicurezza in Microsoft .NET

2.1 Le basi per la sicurezza

Le basi per la sicurezza dell'architettura .NET sono radicate nel CLR e prendono il nome di Esecuzione Gestita.

Durante l'esecuzione gestita il CLR ha la completa conoscenza di tutti gli aspetti del programma in esecuzione. Questo significa conoscere lo stato e la vita delle variabili locali in un metodo. Significa anche conoscere la provenienza del codice per ognuno degli stack frame. Tutto ciò permette di conoscere tutti gli oggetti esistenti ed i riferimenti agli oggetti, incluse le informazioni sulla raggiungibilità.

Il codice gestito viene accuratamente verificato dal Loader degli Assembly per garantirne la correttezza a livello di gestione dei tipi, nonché per accertare il corretto funzionamento di altre proprietà rispetto agli standard definiti.

Questo tipo di protezione rende teoricamente impossibili tentativi di violazione della security con attacchi di tipo buffer overflows o riferimenti non

autorizzati ad aree di memoria private.

Se questo è il meccanismo di base per garantire la cosiddetta Type Safety in applicazioni .NET, bisogna sapere che la politica di protezione è ben più articolata, si avvale di meccanismi di crittografia e firma digitale ed utilizza, inoltre, i seguenti meccanismi di protezione:

1. Evidence Based Security
2. Code Access Security
3. Role Based Security
4. Isolate Storage

2.2 Evidence Based Security

Gli elementi chiave del modello di sicurezza basato sulle prove sono costituiti dalle Evidence (prove dell'assembly), le Policy (politiche di sicurezza) e le Permissions (Autorizzazioni).

2.2.1 Evidence (Prove dell'assembly)

Il termine prove indica semplicemente le informazioni relative al codice passate ai criteri di protezione. Si tratta fondamentalmente delle risposte al gruppo di domande generalmente poste tramite i criteri di protezione:

- Da quale sito proviene l'assembly?
- Da quale URL proviene l'assembly? I criteri di protezione richiedono l'indirizzo specifico da cui è stato scaricato l'assembly.

- Da quale area proviene l'assembly? Le aree sono descrizioni dei criteri di protezione, ad esempio Internet, Intranet, computer locale e così via, basate sulla posizione del codice.

- Qual è il nome sicuro dell'assembly? Il nome sicuro (Strong Name) è un identificatore sicuro dal punto di vista crittografico, fornito dall'autore dell'assembly. Sebbene non rappresenti in effetti un meccanismo di autenticazione dell'autore, identifica in modo univoco l'assembly e garantisce che non sia stato contraffatto.

2.2.2 Policy (Politiche di sicurezza)

Le Policy, fondamentalmente stabiliscono quali risorse possono essere accedute da un assembly, prevenendo che erroneamente, o con intenzione malevole venga corrotta l'integrità del sistema.

La funzione basilare dei criteri di protezione è quella di stabilire una corrispondenza tra le prove dell'assembly ed il sistema di autorizzazioni.

2.2.3 Permissions (Sistema di autorizzazioni)

Le autorizzazioni sono alla base del sistema di protezione. Stabiliscono una corrispondenza tra le risorse ed i diritti di accesso sulla base delle prove ricavata dall'assembly. Il Framework .NET stabilisce permessi per vari oggetti tra cui: FileIO, Environment, FileDialog, Registry, Socket, Web, Isolate Storage etc.

È possibile che l'autore di un assembly sappia che per il corretto funzionamento è necessario un set minimo di autorizzazioni oppure che l'assembly non richiederà mai determinate autorizzazioni. Queste informazioni aggiuntive sui

requisiti possono essere passate al sistema di criteri tramite una o più richieste di autorizzazioni specifiche.

2.3 Code Access Security

Il sistema di protezione di accesso al codice garantisce che il codice di un assembly non tenti di violare i permessi di esecuzione assegnati.

I permessi di esecuzione vengono stabiliti al caricamento di assembly in memoria. Se un metodo dell'assembly tenta di accedere ad una risorsa, viene avviata una procedura di verifica dei diritti di accesso, ispezionando tutta la catena delle chiamate.

2.4 Role Based Security

A volte è opportuno che le decisioni relative alle autorizzazioni vengano basate sull'identità autenticata o sul ruolo associato al contesto di esecuzione del codice. I servizi disponibili in .NET Framework consentono di incorporare questo tipo di logica nelle applicazioni in modo semplice, sulla base dei concetti di identità e principal.

Le identità possono corrispondere all'utente connesso al sistema operativo o essere definite dall'applicazione.

Il principal corrispondente include l'identità oltre a tutte le informazioni eventualmente correlate al ruolo, ad esempio il gruppo dell'utente definito dal sistema operativo.

2.5 Isolate Storage: Spazi di archiviazione isolati

In .NET Framework è disponibile una speciale infrastruttura denominata archiviazione isolata, un meccanismo molto simile alla SandBox di Java, che supporta l'archiviazione dei dati anche nel caso non sia consentito l'accesso ai file.

L'infrastruttura di archiviazione isolata viene implementata tramite un nuovo gruppo di tipi e metodi supportati da .NET Framework per l'archiviazione locale. Sostanzialmente, ad ogni assembly viene concesso l'accesso a uno spazio di archiviazione isolato su disco. Non viene permesso l'accesso ad altri dati e ogni spazio di archiviazione isolato è disponibile esclusivamente per l'assembly specifico per cui è stato creato.

Le applicazioni possono avvalersi di questi spazi di archiviazione isolati per la memorizzazione dei log, il salvataggio delle impostazioni, oppure per il salvataggio su disco dei dati sullo stato per utilizzi successivi. Poiché la posizione dello spazio di archiviazione isolato è predeterminata, questo strumento risulta piuttosto utile per definire spazi di archiviazione univoci senza dover definire percorsi specifici per i file.

Anche per il codice di Intranet locali sono previste restrizioni simili, anche se meno limitative, e per tale codice è disponibile una quota maggiore dello spazio di archiviazione isolato. Infine, il codice proveniente dall'area Siti con restrizioni (ovvero siti considerati non affidabili) non ottiene l'accesso allo spazio di archiviazione isolato.

Parte II

SIF: Flusso sicuro delle informazioni

Capitolo 3

Flusso sicuro delle informazioni

3.1 Introduzione

Nel precedente capitolo è stato affrontato il problema della sicurezza in applicazioni .NET. Si è visto che il modello di sicurezza di Microsoft .NET è un valido strumento di protezione dell'integrità del sistema. Questo modello di sicurezza, tuttavia, non è in grado di garantire controlli sulla propagazione di informazioni private a cui è stato concesso l'accesso.

3.2 Il problema

Le politiche di sicurezza adottate in .NET permettono un meccanismo di controllo degli accessi efficace ed articolato. Tuttavia nel contesto di un applicazione autorizzata all'esecuzione, che esegue solo operazioni consentite, qual'è l'effettivo controllo sulle informazioni manipolate?

Alcune applicazioni hanno la necessità di accedere a files privati, o di mani-

Figura 3.1: Programma per il calcolo del codice fiscale, implementato come controllo gestito

polare informazioni personali per fornire all'utente servizi di utilità, quest'ultimo vorrebbe permettere la lettura dei propri dati, ma impedire la pubblicazione degli stessi.

In questo contesto una politica di sicurezza basata sul controllo degli accessi si rivela inefficace e richiede dei meccanismi di protezione più affidabili, come un analisi statica del codice atta al controllo dei flussi di informazioni.

Per chiarire meglio la problematica consideriamo una semplice utility, realizzata come controllo gestito[1], per il calcolo del Codice Fiscale, vedi Figura 3.1: si tratta di una applicazione molto utile che richiede l'inserimento di molti dati personali, tra cui la data di nascita, il comune di nascita etc.

Con una logica di funzionamento molto simile a quella delle Applet Java, un

[1]Il controllo gestito è l'equivalente delle applet Java, viene eseguito direttamente all'interno della finestra del Browser

controllo gestito viene scaricato sull'host ed eseguito dal CLR in modalità isolata con più o meno diritti di accesso alle risorse, a seconda che il sito provenga dalla Intranet o da Internet. La grande utilità di una gestione di questo tipo risiede nel fatto che il CLR è in grado di avviare l'applicazione in modalità Windowed, ossia completamente integrata con la pagina Web di riferimento. Al contrario di Java, che nei casi più semplici avvisa l'utente della presenza di un'Applet visualizzando nel Tray Icon un'apposita icona, il Framework .NET non dà alcun segno del caricamento di un controllo gestito, questo per il semplice motivo che è assolutamente sicuro di aver relegato il controllo in un area isolata.

3.3 Funzionamento del programma d'esempio

La logica di funzionamento di un programma di calcolo del Codice Fiscale è molto semplice ed è per questo motivo che è stato scelto come esempio per la problematica della tutela della riservatezza delle informazioni.

L'infrastruttura allestita per il funzionamento del programma è mostrata in Figura 3.2 e si avvale dei seguenti componenti:

- una pagina Web, in modalità Client-Side, che ospita il controllo gestito ed è adibita all'interfacciamento con l'utente
- un database remoto, utilizzato per la gestione dei dati utente
- un insieme di applicazioni Server-Side, necessarie per il funzionamento del programma ed altre funzionalità non strettamente necessarie.

I dati inseriti dall'utente vengono inviati al server remoto per il calcolo del

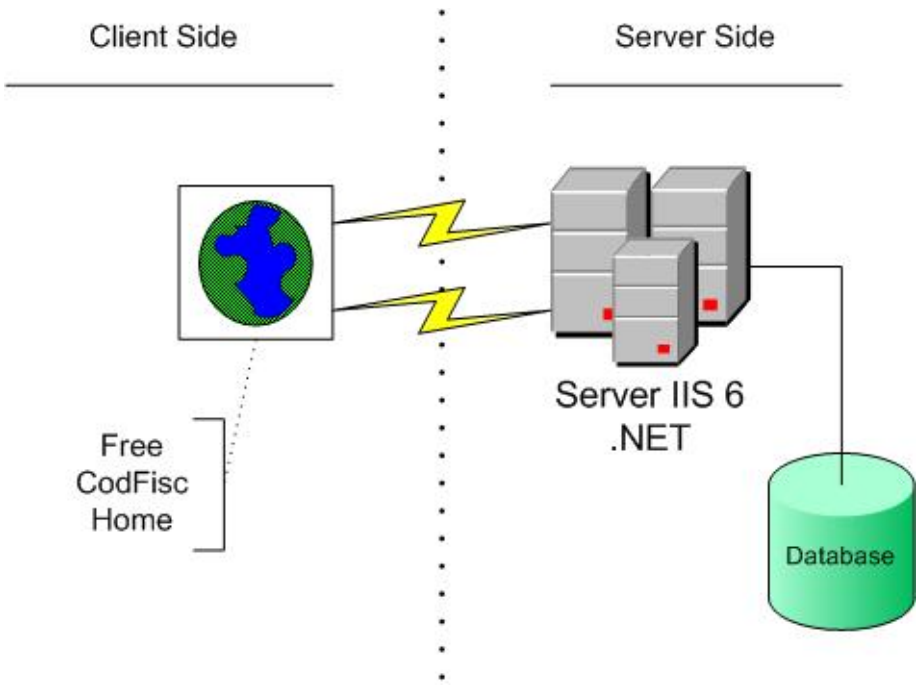

Figura 3.2: Infrastruttura necessaria al funzionamento del programma

Codice Fiscale. Questa tecnica è utilizzata per carpire i dati privati dell'utente, Figura 3.3.

È evidente come l'uso improprio di questa architettura software permetta, in modo semplice e nel totale rispetto dei meccanismi di protezione, una violazione della privacy, consentendo l'invio indiscriminato di informazioni personali. Questo tipo di violazione viene detta **Security Information Flow** (SIF). Un sistema di protezione basato esclusivamente sulle tecnologie precedentemente riassunte, in primis l'archiviazione isolata, non permette di risolvere una situazione di questo tipo. Una possibile soluzione sarebbe quella di restringere, ulteriormente, la politica di sicurezza, impedendo a programmi scaricati dalla rete di accedere ai socket, tuttavia un approccio di questo tipo risulterebbe molto limitativo e non darebbe più senso ad applicazioni di questo tipo.

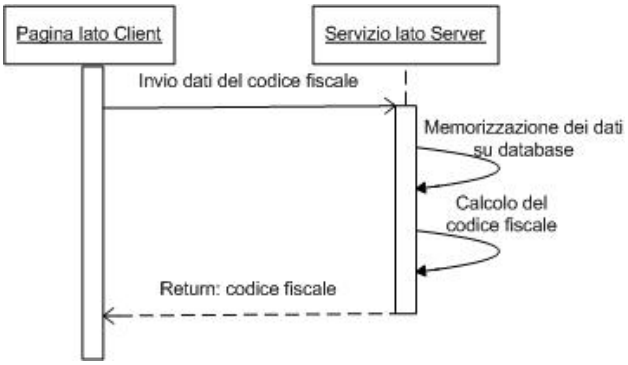

Figura 3.3: I dati, inviati dal client, vengono intercettati dal server remoto, che, per non destare troppo sospetto, ritorna effettivamente il codice fiscale

Si rende quindi necessario l'utilizzo di nuovi meccanismi come l'analisi statica del codice, atti a controllare il flusso delle informazioni. Possiamo, ad esempio, assegnare un alto livello di segretezza alle informazioni ritenute private, ad esempio la struttura dati che contiene i dati dell'utente, ed un basso livello di segretezza alle informazioni inviate sui socket: in questo modo un tentativo di invio di dati personali risulterebbe in un assegnamento di dati ad alto livello di sicurezza verso dati a basso livello di segretezza e quindi ad alto rischio. Un problema di questo tipo sarebbe chiaramente sintomo di una violazione della Privacy.

Quindi, attraverso l'uso di uno strumento capace di rilevare tali flussi di informazione, l'utente incauto può verificare la maliziosità del codice, prima di utilizzarlo.

3.4 SIF: Security Information Flow

Dato un programma ed un assegnamento di livelli di sicurezza alle informazioni a cui accede il programma, il programma soddisfa il Secure Information Flow se,

alla terminazione del programma, informazioni ad un dato livello di sicurezza non si sono propagate in oggetti di livello inferiore.

Lo studio del SIF di un programma viene effettuato analizzando il SIF di ogni singolo metodo, assumendo che, quando verifichiamo un metodo, gli altri metodi soddisfano tale proprietà.

L'information flow nel codice CIL di un metodo può essere esplicito o implicito. Si parla di information flow esplicito nel caso di un assegnamento. Un flusso di informazioni si dice, invece, implicito quando viene usato un valore all'interno di una istruzione condizionale.

Per chiarire meglio quando siamo in presenza di un flusso di informazione implicito, vediamo l'esempio riportato in Figura 3.4: si tratta del codice CIL di un metodo *mt* di una classe A. Supponiamo che il registro `x1` (argomento del metodo A.*mt*) contenga il riferimento di un oggetto di una classe B. Inoltre, assumiamo che il livello di sicurezza della classe B sia più alto di quello della classe A. Bisogna notare che il registro `x0` contiene un riferimento alla classe A. Al termine dell'esecuzione del codice CIL, il valore finale del campo *f1* dell'oggetto A è 0 o 1 a seconda del valore del campo *f2* dell'oggetto B.

Il codice d'esempio illustra la tipica situazione in cui si presenta un caso di flusso implicito di informazione. C'è una violazione perché controllando il valore finale di A.*f1* viene rivelata l'informazione sul valore di B.*f2*, che ha un livello di sicurezza più alto.

In generale si ha flusso implicito all'interno di un'istruzione di salto (if-else, swith, etc.). Quindi, al fine di rilevare i flussi impliciti, è di fondamentale importanza conoscere le dipendenze tra le verie istruzioni. Poichè il CIL è un

```
0 : ldloc  x0
1 : ldarg  x1
2 : ldfld  B.f2
3 : bge    6
4 : ldc.i4 0
5 : jmp    7
6 : ldc.i4 1
7 : stfld  A.f1
8 : ldc.i4 1
9 : ret
```

Figura 3.4: Un caso di flusso implicito in un set di istruzioni CIL

linguaggio non strutturato, è stato necessario realizzare un apposito strumento: il DepComputerIL.

Capitolo 4

DepComputerIL: tool per il calcolo delle dipendenze

4.1 Introduzione

Il DepComputerIL è un tool appositamente realizzato per il calcolo delle dipendenze tra le istruzioni del codice CIL.

Lo strumento è dotato delle seguenti funzionalità:

1. Disassembla un qualunque Assembly .NET, in particolare è in grado di disassemblare file eseguibili e librerie .dll scritti con un qualunque linguaggio .NET (J#, VB.NET, C#, Cobol, Managed C++, etc..).

2. Per ogni metodo dichiarato all'interno di un Tipo, costruisce il relativo Control-Flow Graph.

3. Calcola l'IPD di ciascuna istruzione del metodo.

4. Per ogni istruzione, scrive su file l'offset delle istruzioni da cui questa dipende.

4.2 Set di istruzioni trattate

Il Common Intermediate Language (CIL) è il set di istruzioni object-oriented con cui vengono distribuiti gli Assembly .NET. CIL supporta numerosi costrutti di alto livello, inclusa l'ereditarietà, la garbage collection ed i meccanismi di security e type safety. Sono inclusi poi tutta una serie di costrutti come i puntatori non gestiti ed altre istruzioni che non sono verificabili, ma sono presenti solo per garantire il supporto di compilatori che, palesemente, non sono Type-Safe come quello *C++*.

Noi ci occupiamo solo di una piccola parte del CIL, cioè quella che può provocare flusso implicito. Riportiamo qui di seguito l'elenco di tali istruzioni con la relativa descrizione: 4.1

4.3 Concetti di base

In questo paragrafo verranno esposti alcuni concetti di fondamentale importanza per la comprensione dell'algoritmo utilizzato dal DepComputerIL: *Control-Flow Graph* (CFG), *Immediate Post Dominators* (IPDs) e *Dipendenze di un'istruzione*.

`beq`	Effettua il salto se value1 è uguale a value2.
`bge`	Effettua il salto se value1 è maggiore o uguale a value2.
`bge.un`	Effettua il salto se value1 è maggiore o uguale a value2, quando vengono confrontati interi unsigned o unordered floating point.
`bgt`	Effettua il salto se value1 è maggiore di value2.
`bgt.un`	Effettua il salto se value1 è maggiore di value2, quando vengono confrontati interi unsigned o unordered floating point.
`ble`	Effettua il salto se value1 è minore o uguale a value2.
`ble.un`	Effettua il salto se value1 è minore o uguale a value2, quando vengono confrontati interi unsigned o unordered floating point.
`blt`	Effettua il salto se value1 è minore di value2.
`blt.un`	Effettua il salto se value1 è minore di value2, quando vengono confrontati interi unsigned o unordered floating point.
`bne.un`	Effettua il salto se value1 diverso da value2, quando vengono confrontati interi unsigned o unordered floating point.
`br`	Salto incondizionato.
`call`	Chiama il metodo indicato dal descrittore di metodo passato.
`callvirt`	Chiama un metodo ad associazione tardiva su un oggetto, inserendo il valore restituito dal metodo nello stack di valutazione.
`jmp`	Esce dal metodo corrente e passa a quello specificato.
`brtrue, brfalse`	Provoca il salto se al top dello stack è presente un valore intero pari a 0, o se è un puntatore impostato a null o un riferimento ad oggetto. brtrue fa il lavoro opposto.
`ret`	Ritorna dal metodo corrente. Il valore restituito, se presente, deve trovarsi al top dello stack.
`switch`	Implementa una tabella di passaggio: ha come argomento un insieme di *target* e, a seconda del valore letto dallo stack, passa il controllo ad uno di questi.

Figura 4.1: Istruzioni trattate

4.3.1 Control-Flow Graph

Ogni metodo e, in generale, ogni programma è caratterizzato da un *control-Flow graph*, cioè da un insieme di nodi e archi che costituiscono, appunto, un grafo orientato. Ogni nodo è una istruzione e ogni arco è una transazione tra i nodi connessi. Nel nostro caso la transazione corrisponde alla relazione di sequenzialità nell'esecuzione delle istruzioni: in definitiva, se eseguiamo un percorso sul grafo, la sequenza i nodi che attraversiamo risulta essere una possibile sequenza di esecuzione del programma 4.2.

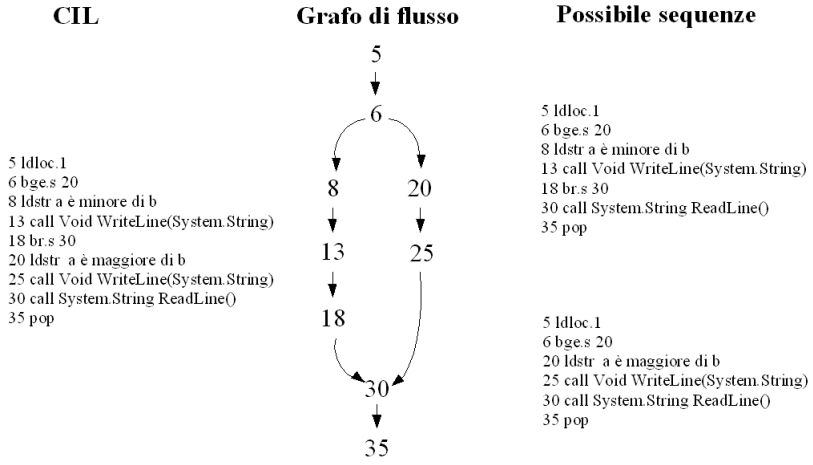

Figura 4.2: Grafo e possibili sequenze di esecuzione

La maggior parte delle istruzioni (nodi) hanno un solo arco entrante ed un solo arco uscente, ma in generale questo non è vero: le istruzioni di salto condizionato, ad esempio, posso avere due o più archi entranti/uscenti, le istruzioni *return*, invece, hanno sempre zero archi uscenti, ma possono avere un numero variabile di archi entranti (ad esempio, perchè sono il target di più istruzioni di salto).

Per comodità di esposizione e di implementazione, abbiamo aggiunto ad ogni

grafo due nodi virtuali: *START* e *END*, i quali caratterizzano, rispettivamente, l'inizio e la fine del metodo 4.3.

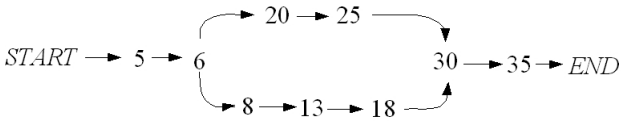

Figura 4.3: Nodi virtuali START e END

Vediamo qualche esempio di *Control-Flow Graph* per alcuni programmi. Per ogni istruzione, il numero di archi uscenti dipende solo dalla particolare istruzioni in questione: in particolare, i salti condizionati (*if*, *switch*) hanno sempre due o più archi uscenti (successori); l'istruzione virtuale *END* è l'unica a non avere alcun arco uscente, mentre tutte le altre ne hanno uno solo. In seguito, gli archi uscenti da una istruzione di salto condizionato li chiameremo *rami* di salto condizionato.

Il numero di archi entranti, invece, è legato a tutta la struttura del grafo e, quindi, del programma. Ogni istruzione ha, solitamente, un solo arco entrante (fatta eccezione per il nodo virtuale *START* che non ha archi entranti), ma, in generale, questo non è vero: infatti, tutte le istruzioni che sono *target* di qualche salto condizionato possono avere più archi entranti 4.4.

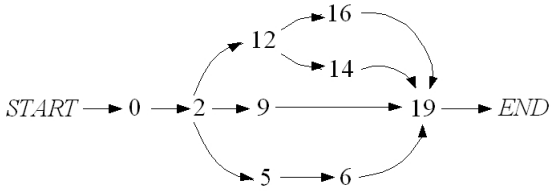

Figura 4.4: Esempio di target con 4 archi entranti

Nella nostra trattazione, consideriamo solo grafi ben strutturati, cioè in cui un salto non è privo di *target* e non ha come destinatario se stesso, e che

soddisfano la condizione per cui, partendo da qualsiasi istruzione, è sempre possibile trovare un percorso che porti al nodo *END*.

4.3.2 Immediate Post Dominators (IPDs)

È un concetto importante nella ricerca delle dipendenze tra le istruzioni di qualsiasi linguaggio non strutturato. Un *post-dominator* è un istruzione che si trova su tutti i possibili percorsi che partono dal nodo *START* ed arrivano al nodo *END*. Quindi, prese due istruzioni *instr1* e *instr2* di un grafo, diremo che *instr2* post-domina *instr1* se tutti i percorsi dal nodo *START* al nodo *END*, passanti per *instr1*, contengono *instr2*. In particolare, diremo che *instr2* è l'*immediate-post-dominator* di *instr1* se *instr2* è il primo *post-dominator* incontrato sul cammino da *instr1* al nodo *END*. Per chiarire meglio le idee, facciamo riferimanto alla figura 4.5 in cui sono stati messi in evidenza sia *immediate-post-dominator* che *post-dominator*.

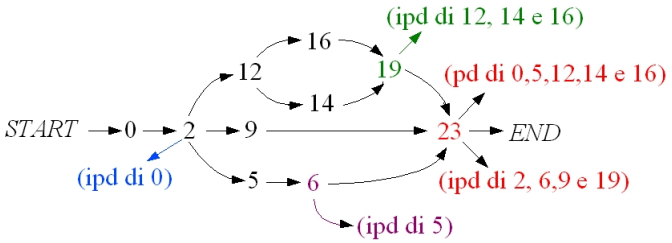

Figura 4.5: Esempio di IPD e PD

4.3.3 Dipendenze di un'istruzione

Vediamo, ora, di capire come si procedere nel calcolare le dipendenze tra le istruzioni del codice CIL, una volta noti gli *IPD* e, in particolare, quelli delle

istruzioni salto. Per far ciò serviamoci del esempio riportato in figura 4.6:

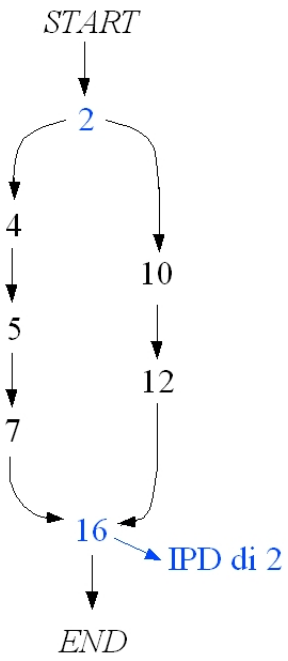

Figura 4.6: Esempio di dipendenza

L'istruzione 2 è un'istruzione di salto (*if-esle*) ed ha come *IPD* l'istruzione 16. Partendo dalla 2, si procede in profondità attraverso i successore di ogni istruzione, fino ad arrivare alla 16. Tutte le istruzioni che si trovano sui percorsi che vanno dalla 2 alla 16 dipendono dalla 2.

Il concetto di *IPD* è, quindi, alla base del calcoo delle dipendenze.

4.4 Funzionamento del tool

Il DepComputerIL è uno strumento molto semplice a livello architetturale: infatti, come mostrato in figura 4.7, è costituito da un solo modulo, il *DepComputer*, che fa uso di una libreria preesistente, la *IlParser.dll*.

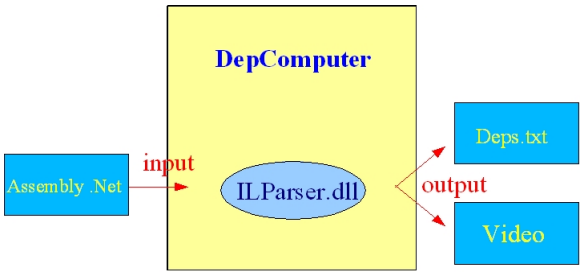

Figura 4.7: Architettura di alto livello

Il funzionamento del *DepCpmputerIL* è schematizzabile in 6 step:

1. **Diassemblaggio**: per effettuare quest'operazione, abbiamo fatto ricorso alla libreria *IlParser.dll*: si tratta di una libreria sviluppata all'interno del nostro dipartimento durante la realizzazione del tool *SIFDotNet*. Questa fa uso a sua volta della libreria *IlReader.dll*, sviluppata da Lutz Roeder all'interno del progetto MONO[1].

2. **Analisi metodo per metodo**: per ogni Tipo dichiarato nell'assembly, si considerano tutti i suoi metodi, uno alla volta. Per ogni metodo si costruisce un *control-flow graph*, in cui ogni nodo corrisponde ad un'istruzione.

3. **Depth-First Search**: su ogni control-flow graph costruito, si effettua una visita anticipata in profondità a partire dal nodo virtuale *END* fino al nodo virtuale *START*, andando a numerare i nodi da 1 a n. Questa numerazione la chiameremo *numerazione DFS*. Attraverso questa operazio-

[1]http://www.go-mono.com/: si tratta di un progetto Open-Source sponsorizzato dalla Novell che ha come scopo quello di creare un'implementazione gratuita di .Net e, soprattutto, portabile su altre piattaforme.

ne, si ottiene una struttura ausiliaria, detta *foresta*, la quale rappresenta l'albero intermedio ottenuto dalla visita anticipata.

4. **Calcolo del IPD**: per ogni istruzione del control-flow graph, viene calcolato l'IPD, sfruttando la numerazione DFS. La computazione, nodo per nodo, viene eseguita seguendo la numerazione DFS in ordine decrescente.

5. **Calcolo delle dipendenze**: si scorre il control-flow graph da *START* a *END* e, sfruttando la conoscenza dell'IPD di ogni nodo, si determinano le dipendenze tra le istruzioni.

6. **Output**: il tool fornisce due tipi di uscita:

   - *uscita a video*: durante la computazione, mostra all'utente le caratterstiche di ogni nodo e le dipendenze trovate.
   - *uscita su file*: le dipendenze trovate vengono scritte sul file di testo *Deps.txt*. Si è scelto di referenziare ogni singola istruzione attraverso il proprio offset, e non mediante l'indice all'interno del codice CIL.

4.5 Dettagli architetturali e implementativi

In questo paragrafo vedremo più in dettaglio la struttura logica della libreria *IlParser.dll* e, soprattutto, quella del *DepComputer*.

4.5.1 ILParser.dll

Come mostra la figura 4.8, all'interno di questa libreria abbiamo tre classi:

- *Parser*
- *ILMethodBody*
- *ILInstruction*

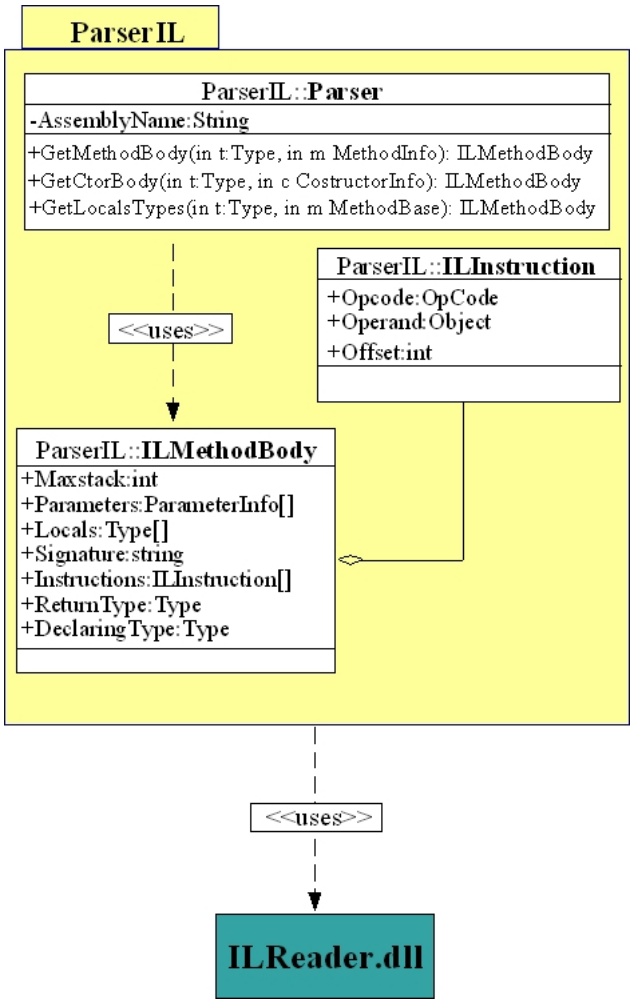

Figura 4.8: ParserIL: sono visibili i Tipi esportati

Per quanto riguarda l'utilizzo da parte del **ParserIL** della libreria ILReader.dll in Figura 4.9 si descrivono le operazioni necessarie per portare a termine

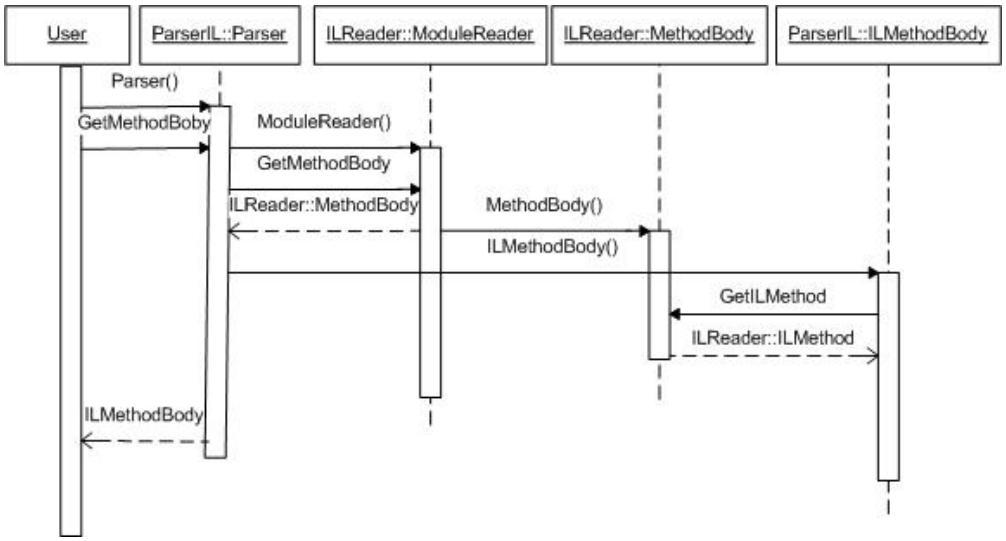

Figura 4.9: ParserIL: Sequenza delle azioni intraprese in seguito ad una richiesta di servizio

la richiesta di un metodo da parte di uno *User*. La libreria mette a disposizione il Tipo *ModuleReader* per effettuare le operazioni di lettura dell'assembly. Utilizzando il metodo *MethodBody.GetMethodBody()* è possibile recuperare il codice CIL del metodo richiesto.

Parser

Il Tipo *Parser* rappresenta il cuore della libreria; esso rende disponibili tre metodi pubblici con i quali è possibile effettuare delle richieste di servizio. Questi sono:

- *ILMethodBody GetMethodBody(Type t, MethodInfo m)* utilizzato per richiedere al Parser il codice CIL di un metodo m, appartenente ad tipo t. Il tipo restituito, infatti, come sarà chiaro più avanti, contiene tutte le informazioni del metodo disassemblato.

- *ILMethodBody GetCtorBody(Type t, ConstructorInfo c)* utilizzato per richiedere il codice CIL di un costruttore c del tipo t.

- *Type[] GetLocalsTypes(Type t, MethodBase m)* utilizzato se sono necessarie soltanto le variabili locali di un metodo m, appartenente al tipo t.

I metodi *GetMethodBody e GetCtorBody* restituiscono un Tipo di nome *ILMethodBody*.

ILMethodBody

Rappresenta la struttura di un metodo CIL, la quale viene rappresentata in questo modo:

- *MaxStack*: Dimensione massima dell'evaluation stack

- *Parameters*: Array di Tipi che rappresentano gli argomenti del metodo

- *Locals*: Array di Tipi che rappresentano le variabili locali

- *Signature*: Firma del metodo

- *Instructions*: Array di ILInstruction che rappresenta l'insieme di istruzioni CIL del metodo

- *ReturnType*: Tipo ritornato dal metodo

- *DeclaringType*: Tipo proprietario del metodo

ILInstruction

Un'istruzione CIL viene rappresentata mediante il Tipo *ILInstruction*, il quale esporta tre informazioni:

1. *Opcode*: Opcode dell'istruzione CIL
2. *Operand*: Operando
3. *Offset*: Offset dell'istruzione all'interno del metodo

4.5.2 DepComputerIL

Come detto in precedenza, lo scopo del nostro tool è quello di calcolare le dipendenze tra le istruzioni di un qualsiasi codice CIL. Per svolgere questa mansione, il sistema è stato implementato attraverso tre classi:

- *DepComputer*
- *CFG*
- *Node*

Vediamole in modo più approfondito, una ad una.

Node

Il Tipo *Node* rappresenta la singola istruzione all'interno del codice CIL ed ha una struttura interna abbastanza semplice, come mostrata in figura 4.10.

Il costruttore *Node()* inizializza tutte le strutture dati della parte privata. Di particolare interesse è il tipo predefinito *ArrayList*: questo implementa una

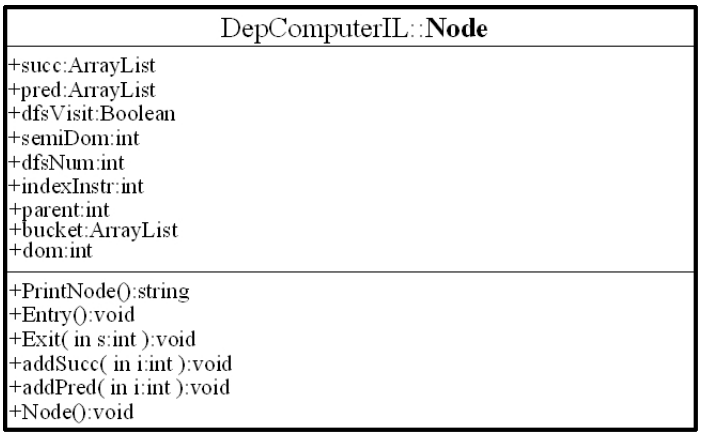

Figura 4.10: Struttura della classe Node

struttura dinamica, simile ad una lista, di tipo *object*, cioè il tipo più astratto in modo che, attraverso un *cast* esplicito, l'utente possa utillizarla a suo piacimento. Nel nostro caso il *cast* è stato utillizzato per gestire oggetti di tipi *int*. Il Tipo *ArrayList* ha, inoltre, una serie di metodi predefiniti che rendono il suo utilizza da parte dell'utente ancora più semplice. Per aumentare la leggibilità del codice, sono stati implementati dei metodi che si limitano a richiamare quelli predefiniti:

- *Entry()*: inizializza il nodo *START*, inserendo tra suoi successori la prima istruzione del codice CIL;
- *Exit(int s)*: inserisce il nodo di indice s tra i predecessori del nodo virtuale *END*;
- *addSucc(int i)*: aggiunge il nodo di indice i tra i successori del nodo su cui viene chiamato;
- *addPred(int i)*: aggiunge il nodo di indice i tra i predecessori del nodo su cui viene chiamato;

Il metodo *PrintNode()* serve per stampare a video le caratteristiche della singola istruzione: in particolare, fornisce informazione all'utente circa il numero assegnato dalla numerazione DFS e gli indici delle istruzioni precedenti e successive.

Abbiamo poi una serie di proprietà che consentono di accedere in lettura e in scrittura alle strutture dati della parte privata: questo è necessario per il funzionamento dell'algoritmo di ricerca dell'IPD.

CFG

Il Tipo **CFG** rappresenta il control-flow graph costruito a partire dal codice CIL del metodo di cui si vogliono calcolare le dipendenze. La figura 4.11 mostra la struttura interna di questo Tipo.

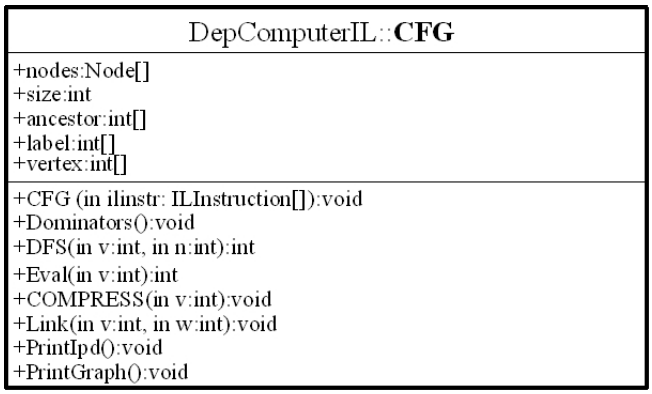

Figura 4.11: Struttura della classe CFG

Il costruttore *CFG (ILInstruction[] ilinstr)* prede in ingresso un array di *ILInstruction*, crea un array di tipo *Node* di dimensioni appropriate e calcola, per ogni nodo del grafo, i successori e i predecessori per poi memorizzarli nelle

apposite strutture dati. Oltre a questo, inizializza le strutture dati private necessarie per il funzionamento dell'algoritmo.

Abbiamo poi i metodi che implementano l'algoritmo vero e proprio:

- *DFS()*: effettua la visita anticipata in profondità sul control-flow graph;
- *Dominators()*: calcola gli IPD di ogni nodo;
- *Link(int v, int w)*: inserisce delle informazioni nella *foresta*; in particolare, vi aggiunge l'arco (v,w);
- *Eval(int v)*: estrae delle informazioni dalla *foresta*; in particolare ritorna l'indice di un nodo;
- *COMPRESS(int v)*: effettua la compressione della porzione di grafo da analizzare;

Abbiamo poi implementato i metodi *PrintIpd()* e *PrintGraph()*, il cui unico scopo è quello di consentire l'uscita a video durante la computazione.

DepComputer

Come sottolinea anche il nome, questo Tipo è la parte centrale di tutto il nostro tool: infatti, ha lo scopo di calcolare le dipendenze e di fornire sia l'uscita a video che quella su file. In figura 4.12 abbiamo riportato la sua struttura interna.

Per effettuare l'output su file, ci serviamo del Tipo predefinito *StreamWriter*: questo implementa un canale di comunicazione tra il nostro tool e il file su cui vogliamo effettuare l'output. La comunicazione è possibile attraverso l'uso dei vari metodi di cui questo Tipo è dotato.

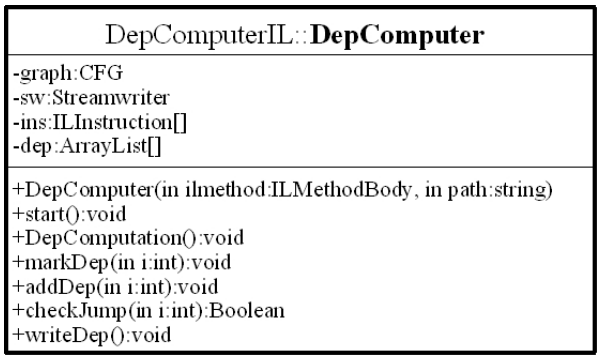

Figura 4.12: Struttura della classe DepComputer

Il costruttore *DepComputer(ILMethodBody ilmethod, string path)* prende in ingresso un oggetto di tipo *ILMethodBody* ed uno di tipo *string*: il primo contiene tutte le informazioni, ottenute in fase di parser, circa il metodo da analizzare, mentre il secondo specifica il nome del file su cui vogliamo l'output.

Abbiamo poi i metodi per il calcolo delle dipendenze:

- *DepComputation()*: per ogni istruzione di salto, invoca la *markDep(int i)*, dove i è l'indice dell'istruzione di salto;
- *markDep(int i)*: chiama la *addDep(int i, int s)* su ogni successore s di i;
- *addDep(int i, int s)*: aggiunge i alla lista delle dipendenze dell'istruzione su ci è stata chiamata e, ricorsivamente, prosegue in profondità fino a raggiungere o l'IPD o il nodo virtuale *END*.

Per finire abbiamo il metodo *writeDep()* il quale ha lo scopo di scrivere su file, attraverso l'uso di un oggetto di tipo *StreamWriter*, le dipendenze precedentemente calcolate.

Capitolo 5

Test svolti sul DepComputerIL

Vediamo ora una serie di casi significativi per valutare il funzionamento del nostro tool. La fase di test è stata effettuata partendo da casi abbastanza semplici, quali istruzioni *if-else* e *cicli*, fino ad arrivare a casi più complessi, quali *if-else* nidificati in *cicli* o in *switch* o entrambi e viceversa.

5.1 Due casi semplici

5.1.1 Ciclo

Cominciamo col vedere il codice sorgente del nostro programma di test:

```
namespace Ciclo
{
class Test
{
static void Main(string[] args)
{
int a = 10;
while (a > 0)
{
Console.WriteLine("Il valore di a è {0}", a);
a--;
}
Console.ReadLine();
```

```
}
}
}
```

Attraverso la funzionalità di diassemblaggio del DepComputerIL, otteniamo il seguente codice CIL, a quale corrisponde il grafo riportato in figura 5.1:

```
0 ldc.i4.s 10
2 stloc.0
3 br.s 25
5 ldstr Il valore di a è {0}
10 ldloc.0
11 box System.Int32
16 call Void WriteLine(System.String, System.Object)
21 ldloc.0
22 ldc.i4.1
23 sub
24 stloc.0
25 ldloc.0
26 ldc.i4.0
27 bgt.s 5
29 call System.String ReadLine()
34 pop
35 ret
```

Non ci resta, quindi, che vedere cosa viene fornito in uscita dal nostro tool: per prima riportiamo l'uscita a video e, a seguire, quella su file.

```
Void Main(System.String[]) :

0 ldc.i4.s 10
2 stloc.0
3 br.s 25
5 ldstr Il valore di a  {0}
10 ldloc.0
11 box System.Int32
16 call Void WriteLine(System.String, System.Object)
21 ldloc.0
22 ldc.i4.1
23 sub
24 stloc.0
25 ldloc.0
26 ldc.i4.0
27 bgt.s 5
29 call System.String ReadLine()
34 pop
35 ret

 Analisi di un metodo:
---------------------------------------
Il nodo 1 ha i seguenti parametri:
DFSNum: 11
il nodo 1 ha 1 successori:
2
```

```
il nodo 1 ha 0 predecessori:

---------------------------------------
Il nodo 2 ha i seguenti parametri:
DFSNum: 10
il nodo 2 ha 1 successori:
3
il nodo 2 ha 1 predecessori:
1

---------------------------------------
Il nodo 3 ha i seguenti parametri:
DFSNum: 9
il nodo 3 ha 1 successori:
4
il nodo 3 ha 1 predecessori:
2

---------------------------------------
Il nodo 4 ha i seguenti parametri:
DFSNum: 8
il nodo 4 ha 1 successori:
13
il nodo 4 ha 1 predecessori:
3

---------------------------------------
Il nodo 5 ha i seguenti parametri:
DFSNum: 19
il nodo 5 ha 1 successori:
6
il nodo 5 ha 1 predecessori:
15

---------------------------------------
Il nodo 6 ha i seguenti parametri:
DFSNum: 18
il nodo 6 ha 1 successori:
7
il nodo 6 ha 1 predecessori:
5

---------------------------------------
Il nodo 7 ha i seguenti parametri:
DFSNum: 17
il nodo 7 ha 1 successori:
8
il nodo 7 ha 1 predecessori:
6

---------------------------------------
Il nodo 8 ha i seguenti parametri:
DFSNum: 16
il nodo 8 ha 1 successori:
9
il nodo 8 ha 1 predecessori:
7

---------------------------------------
Il nodo 9 ha i seguenti parametri:
DFSNum: 15
```

```
il nodo 9 ha 1 successori:
10
il nodo 9 ha 1 predecessori:
8

---------------------------------------
Il nodo 10 ha i seguenti parametri:
DFSNum: 14
il nodo 10 ha 1 successori:
11
il nodo 10 ha 1 predecessori:
9

---------------------------------------
Il nodo 11 ha i seguenti parametri:
DFSNum: 13
il nodo 11 ha 1 successori:
12
il nodo 11 ha 1 predecessori:
10

---------------------------------------
Il nodo 12 ha i seguenti parametri:
DFSNum: 12
il nodo 12 ha 1 successori:
13
il nodo 12 ha 1 predecessori:
11

---------------------------------------
Il nodo 13 ha i seguenti parametri:
DFSNum: 7
il nodo 13 ha 1 successori:
14
il nodo 13 ha 2 predecessori:
4
12

---------------------------------------
Il nodo 14 ha i seguenti parametri:
DFSNum: 6
il nodo 14 ha 1 successori:
15
il nodo 14 ha 1 predecessori:
13

---------------------------------------
Il nodo 15 ha i seguenti parametri:
DFSNum: 5
il nodo 15 ha 2 successori:
16
5
il nodo 15 ha 1 predecessori:
14

---------------------------------------
Il nodo 16 ha i seguenti parametri:
DFSNum: 4
il nodo 16 ha 1 successori:
17
il nodo 16 ha 1 predecessori:
```

```
15

---------------------------------------
Il nodo 17 ha i seguenti parametri:
DFSNum: 3
il nodo 17 ha 1 successori:
18
il nodo 17 ha 1 predecessori:
16

---------------------------------------
Il nodo 18 ha i seguenti parametri:
DFSNum: 2
il nodo 18 ha 1 successori:
19
il nodo 18 ha 1 predecessori:
17

---------------------------------------
Il nodo 19 ha i seguenti parametri:
DFSNum: 1
il nodo 19 ha 0 successori:
il nodo 19 ha 1 predecessori:
18

---------------------------------------
Il nodo 19 ha come ipd 0
Il nodo 18 ha come ipd 19
Il nodo 17 ha come ipd 18
Il nodo 16 ha come ipd 17
Il nodo 15 ha come ipd 16
Il nodo 14 ha come ipd 15
Il nodo 13 ha come ipd 14
Il nodo 12 ha come ipd 13
Il nodo 11 ha come ipd 12
Il nodo 10 ha come ipd 11
Il nodo 9 ha come ipd 10
Il nodo 8 ha come ipd 9
Il nodo 7 ha come ipd 8
Il nodo 6 ha come ipd 7
Il nodo 5 ha come ipd 6
Il nodo 4 ha come ipd 13
Il nodo 3 ha come ipd 4
Il nodo 2 ha come ipd 3
Il nodo 1 ha come ipd 2
---------------------------------------

Inizio il calcolo delle dipendenze...

Aggiungo la dipendenza (3,13)
Aggiungo la dipendenza (4,13)
Aggiungo la dipendenza (5,13)
Aggiungo la dipendenza (6,13)
Aggiungo la dipendenza (7,13)
Aggiungo la dipendenza (8,13)
Aggiungo la dipendenza (9,13)
Aggiungo la dipendenza (10,13)
Aggiungo la dipendenza (11,13)
Aggiungo la dipendenza (12,13)
Aggiungo la dipendenza (13,13)
---------------------------------------
```

```
Aggiungo la dipendenza (0,0)
Aggiungo la dipendenza (1,0)
Aggiungo la dipendenza (2,0)
Aggiungo la dipendenza (3,0)
Aggiungo la dipendenza (3,13)
Aggiungo la dipendenza (4,0)
Aggiungo la dipendenza (4,13)
Aggiungo la dipendenza (5,0)
Aggiungo la dipendenza (5,13)
Aggiungo la dipendenza (6,0)
Aggiungo la dipendenza (6,13)
Aggiungo la dipendenza (7,0)
Aggiungo la dipendenza (7,13)
Aggiungo la dipendenza (8,0)
Aggiungo la dipendenza (8,13)
Aggiungo la dipendenza (9,0)
Aggiungo la dipendenza (9,13)
Aggiungo la dipendenza (10,0)
Aggiungo la dipendenza (10,13)
Aggiungo la dipendenza (11,0)
Aggiungo la dipendenza (11,13)
Aggiungo la dipendenza (12,0)
Aggiungo la dipendenza (12,13)
Aggiungo la dipendenza (13,0)
Aggiungo la dipendenza (13,13)
Aggiungo la dipendenza (14,0)
Aggiungo la dipendenza (15,0)
Aggiungo la dipendenza (16,0)

0
0
0
0,27
0,27
0,27
0,27
0,27
0,27
0,27
0,27
0,27
0,27
0,27
0
0
0
```

5.1.2 If-Else

Il codice sorgente è il seguente:

```
using System;

namespace If_Else
{
```

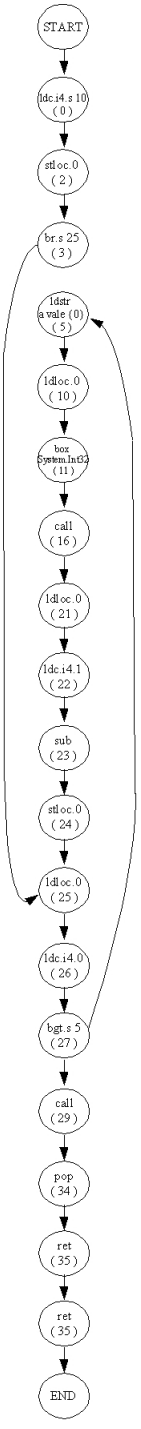

Figura 5.1: Control-flow graph di un ciclo

```
class Test
{
static void Main(string[] args)
{
int a = 3;
int b = 4;
if( a > b )
Console.WriteLine("a è maggiore di b");
else
Console.WriteLine("b è maggiore di a");
Console.ReadLine();
}
}
}
```

Attraverso il diasseblaggio si ottiene il seguente codice CIL, a cui corrisponde il grafo riportato in figura 5.2:

```
0 ldc.i4.3
1 stloc.0
2 ldc.i4.4
3 stloc.1
4 ldloc.0
5 ldloc.1
6 ble.s 20
8 ldstr a è maggiore di b
13 call Void WriteLine(System.String)
18 br.s 30
20 ldstr b è maggiore di a
25 call Void WriteLine(System.String)
30 call System.String ReadLine()
35 pop
36 ret
```

Non riportiamo l'uscita a video, la quale mostra le caratteristiche di ogni nodo del control-flow graph (in particolare, numero DFS, predecessori ,successori e relativo IPD), ma ci limitiamo a l'uscita su file:

```
0
0
0
0
0
0
0
0,6
0,6
0,6
0,6
0,6
0
0
0
```

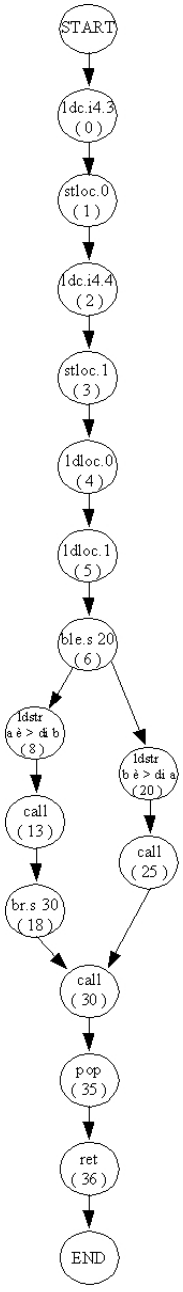

Figura 5.2: Control-flow graph di un'istruzione if-else

5.2 Un caso più complesso

Prendiamo ora in esame un caso più complesso, che combina più istruzioni di salto, come switch, if-else e cicli, annidati tra loro. Come sempre, partiamo dal codice sorgente:

```
using System;

namespace Complesso
{
class Test
{
static void Main(string[] args)
{
int a = 10;
while ( a > 0 )
{
switch( (a%2) )
{
case 0:
Console.WriteLine("a è pari");
if( a == 10)
Console.WriteLine("E' la prima iterazione");
else
Console.WriteLine("Non è la prima iterazione");
break;
case 1:
Console.WriteLine("a è dispari");
if ( a == 9 )
Console.WriteLine("E' la seconda iterazione");
else
Console.WriteLine("Non è la seconda iterazione");
break;
}
a--;
Console.WriteLine();
}
Console.ReadLine();
}
}
}
```

Ancora una volta, fruttiamo la capacità del DepComputerIl per determinare il codice CIL e riportiamo il corrispondente grafo in figura 5.3.

```
Void Main(System.String[]) :

0 ldc.i4.s 10
2 stloc.0
3 br.s 112
5 ldloc.0
6 ldc.i4.2
```

```
7 rem
8 stloc.1
9 ldloc.1
10 switch System.Int32[]
23 br.s 103
25 ldstr a  pari
30 call Void WriteLine(System.String)
35 ldloc.0
36 ldc.i4.s 10
38 bne.un.s 52
40 ldstr E' la prima iterazione
45 call Void WriteLine(System.String)
50 br.s 62
52 ldstr Non  la prima iterazione
57 call Void WriteLine(System.String)
62 br.s 103
64 ldstr a  dispari
69 call Void WriteLine(System.String)
74 ldloc.0
75 ldc.i4.s 9
77 bne.un.s 91
79 ldstr E' la seconda iterazione
84 call Void WriteLine(System.String)
89 br.s 101
91 ldstr Non  la seconda iterazione
96 call Void WriteLine(System.String)
101 br.s 103
103 ldloc.0
104 ldc.i4.1
105 sub
106 stloc.0
107 call Void WriteLine()
112 ldloc.0
113 ldc.i4.0
114 bgt.s 5
116 call System.String ReadLine()
121 pop
122 ret
```

Ancora una volta, ci limitiamo a riportare solo l'uscita su file, in quanto quella a video fornisce le stesse informazioni fornite nei casi precedentemente visti:

```
0
0
0
0,114
0,114
0,114
0,114
0,114
0,114
0,10,114
0,10,114
0,10,114
0,10,114
```

```
0,10,114
0,10,114
0,10,38,114
0,10,38,114
0,10,38,114
0,10,38,114
0,10,38,114
0,10,114
0,10,114
0,10,114
0,10,114
0,10,114
0,10,114
0,10,77,114
0,10,77,114
0,10,77,114
0,10,77,114
0,10,77,114
0,10,114
0,114
0,114
0,114
0,114
0,114
0,114
0,114
0,114
0
0
0
```

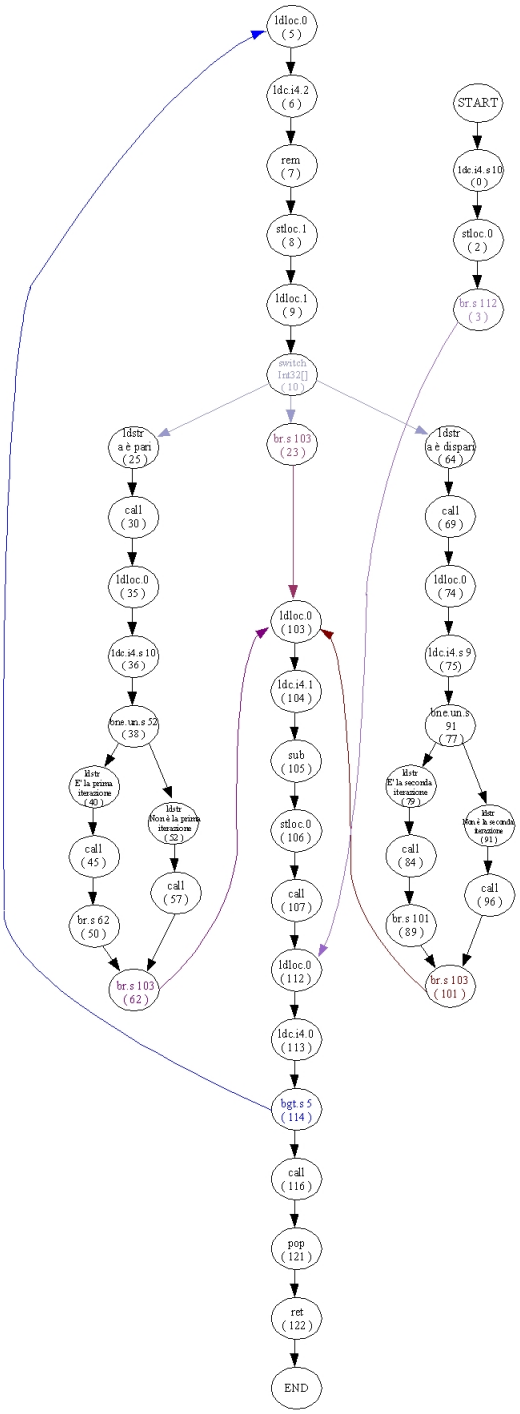

Figura 5.3: Grafo del caso complesso

Parte III

Verifica del flusso di informazione sicuro: ESIFDotNet

Capitolo 6

ESIFDotNet: tool per la tutela delle informazioni private

6.1 Introduzione

SIFDotNet Esteso, chiamato *ESIFDotNet* nel seguito, è un tool di analisi statica del flusso di informazione sicuro per Microsoft .Net, derivato da uno strumento precedentemente realizzato all'interno del nostro dipartimento, il SIFDotNet[1]. Questo tool utilizza un approccio per la verifica della sicurezza delle informazioni sviluppato in [2] e le esperienze maturate nell'ambito dell'analisi statica del codice in ambiente Java.

6.2 Il SIFDotNet

Il SIFDotNet è stato concepito per effettuare un'analisi statica su Assembly .Net e ricercare eventuali violazioni della privacy. Per adampiere a questi scopi,

è stato dotato delle suguenti funzionalità:

- Disassembla un qualunque Assembly .NET, in particolare è in grado di disassemblare file eseguibili e librerie .dll scritti con un qualunque linguaggio .NET (J#, VB.NET, C#, Cobol, Managed C++, etc..)
- Fornisce un **Report** dettagliato della struttura dell'Assembly .NET disassemblato, descrivendo le informazioni in formato XML
- Effettua l'analisi con il metodo del Secure Information Flow, testando ogni metodo, dichiarato all'interno di un Tipo, alla ricerca di una violazione della riservatezza delle informazioni.
- Rileva violazioni della riservatezza delle informazioni, fornendo un rapporto dettagliato delle istruzioni utilizzate per carpire i dati personali.

Questo strumento, quindi, riceve in ingresso un Assembly .Net, ne ricava la struttura interna e la scrive su un file XML. Questo file, detto *securityFile*, viene usato dall'utente per impostare i livelli di sicurezza dei tipi utilizzati nel Assembly. Assegnati i livelli di sicurezza, ha inizio l'analisi vera e propria per la ricerca di violazioni della segretezza delle informazioni.

Intuitivamente, la presenza di un flusso di informazioni illecito può essere rilevato richiedendo che le informazioni con un determinato livello non fluiscano a livelli inferiori. Quindi, dato un programma in cui a ciascuna variabile è assegnato un livello di sicurezza, esso gode della proprietà di flusso sicuro di informazioni se, alla fine della sua esecuzione, il valore di ogni variabile non dipende dal valore iniziale delle variabili con livello di sicurezza superiore.

6.2.1 Struttura interna

Il SIFDotNet è composto da tre sottosistemi:

1. **Sottosistema "Parser"**: ha lo scopo di fare da interfaccia tra l'Assembly e il sottosistema "Abstract Interpreter". Riceve in input un Assembly .NET e, su richiesta, è in grado di fornire il codice CIL di un particolare metodo. Il suo compito è quello di inglobare tutta la logica necessaria per disassemblare un file ed agevolare, quindi, il compito degli altri sottosistemi.

2. **Sottosistema "Security Manager"**: svolge le seguenti funzioni

   - Effettua l'output su file della struttura di un Assembly
   - Si occupa di gestire l'input/output dei livelli di sicurezza con l'operatore
   - Fornisce al sottosistema Abstract Iterpreter il livello di sicurezza dell'elemento richiesto

   Il suo compito richiede la capacità di analizzare la struttura di un Assembly, quindi i Tipi dichiarati ed i rispettivi metodi, con argomenti e variabili locali. Questa funzionalità non richiede la conoscenza del codice CIL, per questo motivo le chiamate al sottosistema Parser sono molto limitate

3. **Sottosistema "Abstract Interpreter"**: Il sottosistema Abstract Interpreter rappresenta il *cuore* di SIFDotNet. Suo compito è analizzare l'assembly fornito dall'operatore, metodo per metodo, alla ricerca di una

violazione della segretezza delle informazioni. L'interprete astratto ha anche il compito di gestire l'output delle informazioni con l'operatore, per comunicare eventuali violazioni e fornire indicazioni circa il metodo e l'istruzione che ha tentato di manipolare in modo scorretto dati ad alto livello di segretezza. Il sottosistema si interfaccia con il *Parser* per richiedere il codice CIL di un metodo ed con il *Security Manager* per richiedere il livello di sicurezza di un Tipo.

6.3 Dal SIFDotNet al ESIFDotNet

Rispetto al tool preesistente, il ESIFDotNet ha due funzionalità in più:

- Calcola delle dipendenze tra le istruzione del codice CIL di ogni metodo
- Effettua un'analisi iterativa del flusso di informazione sicuro

La prima funzionalità è resa possibile grazie all'utilizzo di una versione ridotta del DepComputerIL: infatti, il SIFDotNet era già in grado di diassemblare un Assembly .Net, quindi abbiamo eliminato questa funzionalità dal DepComputerIL, il quale, almeno in questo contesto, riceve direttamente in ingresso un oggetto di tipo *ILMethodBody*. È un'evoluzione molto importante perchè consente la rilevazione dei flussi di informazione impliciti.

L'altra evoluzione del ESIFDotNet è l'iteratività dell'analisi del flusso di informazione sicuro: infatti, una volta che l'utente ha assegnato i livelli di sicurezza ad ogni tipo e ne ha specificato la modificabilità, lo strumento esegue un'analisi iterativa, andando ad aggiornare il *securityFile* ad ogni passo, fino a raggiungere una delle seguenti conclusioni:

- Assembly Sicuro: quando non vi è fuga di informazione;
- Assembly Insicura: quando vi è fuga di informazione;
- Errore nel assegnare i livelli di sicurezza: questo si verifica se l'utente assegna valori di sicurezza e di modificabilità che, potenzialmente, possono essere incoerenti tra loro. Si tratta di una specie di campanello dall'allarme per l'utente, il quale è portato a rivedere i livelli di sicurezza da lui assegnati per capire se ha commesso un errore o meno.

6.4 L'architettura di alto livello

ESIFDotNet, come illustrato in Figura 6.1, è, quindi, composto da quattro sottosistemi:

- Sottosistema "Parser"
- Sottosistema "Security Manager"
- Sottosistema "Abstract Interpreter"
- Sottosistema "DepComputerIL"

Come si vede dalla figura, abbiamo una forte interazione tra l'*Abstract Interpreter* e il *DepComputerIL*: infatti, per ogni metodo da analizzare, l'interprete astratto sfrutta il DepComputerIL per il calcolo delle dipendenze. Questa interazione è necessario al interprete astratto in fase di inizializzazione del ambiente di esecuzione dl metodo da analizzare. Il DepComputer riceve in ingresso un oggetto di tipo *ILMethodBody* e, una volta calcolate le dipendenze, ritorno al interprete astratto il file *deps.txt* contenente le dipendenze.

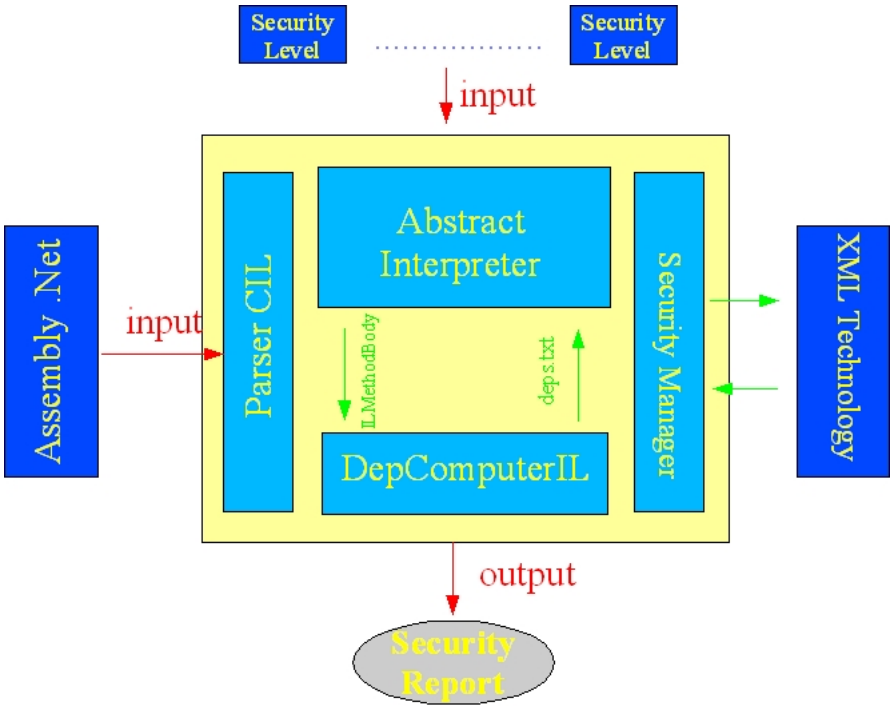

Figura 6.1: Architettura del ESIFDotNet

6.5 Dettagli architetturali e implementativi

La struttura intera del *Parser*, § 4.5.1, e quella del *DepComputerIL*, § 4.5.2, le abbiamo già viste in precedenza. Quindi andremo a vedere più in dettaglio i sottosistemi *Abstract Interpreter* e *Security Manager* e, in particolare, andremo ad analizzare le modifiche fatte rispetto ai relativi sottosistemi del SIFDotNet.

6.5.1 Security Manager

È il sottosistema adibito alla gestione dei livelli di sicurezza ed la sua struttura interna è riportata in figura 6.2.

I Tipo *SecWriter* deve essere istanziato per richiedere al **SecurityManager** di effettuare il *Dump* dell'Assembly (operazione *DumpAssemblySchema*), cioè

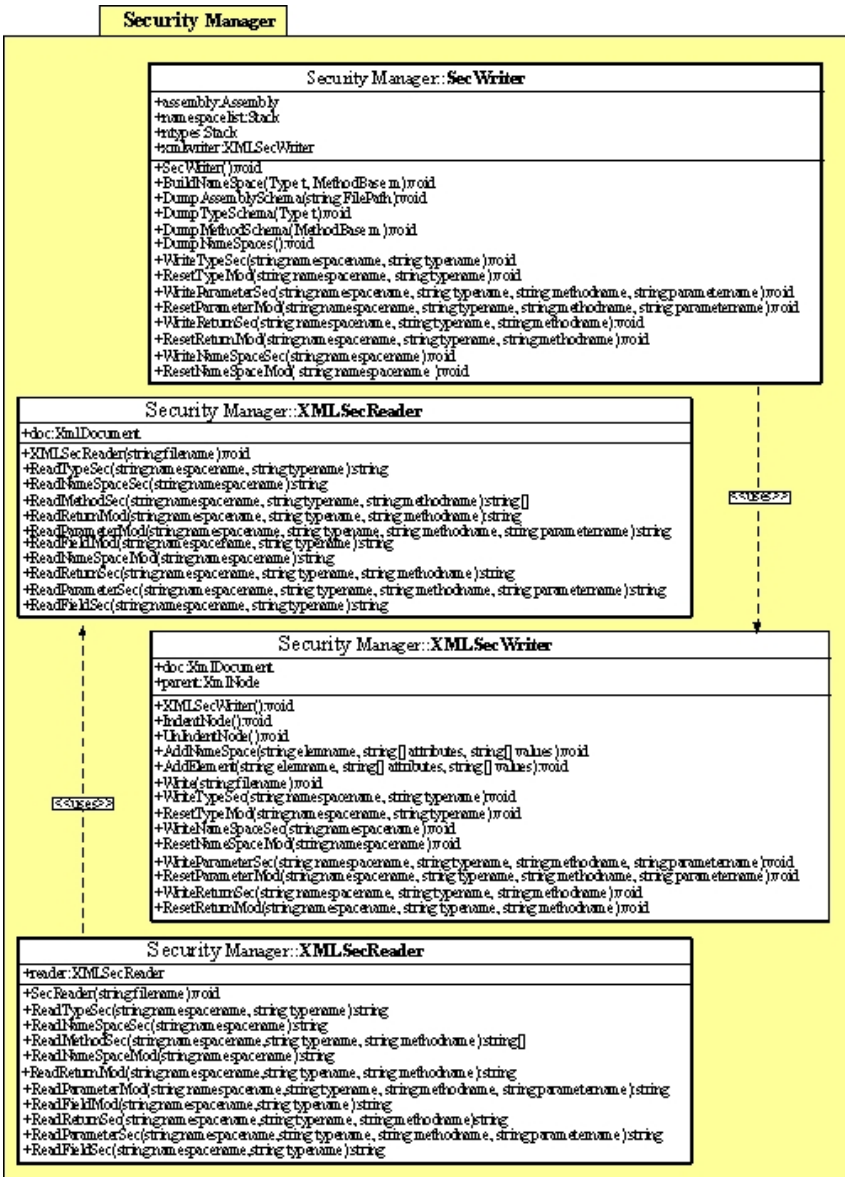

Figura 6.2: Struttura del sottosistema Security manager

scrive su file la struttura dell'Assembly, utilizzando, per la descrizione delle informazioni, una semantica di tipo XML. Il *SecWriter* è in grado di ricavare la struttura di un Assembly senza ricorrere al *Parser*, o dover conoscere i dettagli del codice CIL: infatti, sfrutta la capacità di "riflettere" su un altro Assembly o su se stesso, nota come **Reflection**. Così facendo riesce a ricavare quali Tipi sono dichiarati all'interno di un altro Assembly, quali i metodi, i parametri etc..

Oltre a creare il file XML contenente la struttura dell' Assembly, il *SecWriter* è stato dotato di una serie di metodi per la modifica sia dei livelli di sicurezza che della relativa modificabilità, i quali risultano di fondamentale importanza per l'iteratività dell'analisi:

- *WriteTypeSec* e *ResetTypeMod* servono per accedere al livello di sicurezza di un Tipo e la sua modificabilità. Entrambi questi metodi richiedono il nome del Tipo e il nome del Namespace di appartenenza.

- *WriteParameterSec* e *ResetParameterMod* consentono di lavorare sul livello di sicurezza di un parametro di un metodo e la sua modificabilità. Entrambi richiedono il nome del parametro, del metodo, del Tipo proprietario e del Namespace al quale ilTipo appartiene.

- *WriteReturnSec* e *ResetReturnMod* permettono di andare a modificare il livello di sicurezza del valore di ritorno di un metodo e la sua modificabilità. Entrambi richiedono il nome del Namespace, del Tipo e del metodo.

- *WriteNameSpaceSec* e *ResetNameSpaceMod* servono per accedere al livel-

lo di sicurezza di un Name space. È richiesto solo il nome del Namespace di interesse.

Il Tipo *SecWriter* è, fondamentalmente, solo un'interfaccia tra l'esterno e il Tipo *XMLSecWriter*, il quale, attraverso metodi con lo stesso nome e le stesse caratteristiche di quelli appena visti, è in grado di accedere fisicamente al file XML e eseguire le operazioni necessarie per il funzionamento del nostro tool.

Il SecurityManager fornisce anche la possibilità di leggere il *SecFile* attraverso il Tipo *SecReader*. Anche questo Tipo è un'interfaccia tra l'esterno e un Tipo più vicino al file, il Tipo *XMLSecReader*. Questi due Tipi sfruttano, se pur su livelli diversi, metodi con lo stesso nome e le stesse caratteristiche di utilizzo:

- *ReadTypeSec* e *ReadFieldMod* consentono di leggere il livello di sicurezza di un Tipo e la sua modificabilità. Entrami richedono il nome del Tipo e del Namespace.

- *ReadNameSpaceSec* e *ReadNameSpaceMod*: permettono di leggere illivello di sicurezza di un Namespace e la sua modificabilitàà. Per enrambi è sufficiente il nome del namespace di interesse.

- *ReadMethodSec*, *ReadParameterMod* e *ReadReturnMod*: il primo serve per leggere i livelli di sicurezza dei parametri e del ritorno di un metodo, mentre gli altri due servono per leggere la modificabilità del livello di sicurezza di un particolare parametro o del ritorno del metodo. Tutti e tre richiedono il nome del metodo, del Tipo e del Namespace; il metod ReadParameterMod richiede anche il nome del parametro di interesse.

6.5.2 Abstract Interpreter

Questo sottosistema, essendo la parte principale del nostro tool, è estremamente complesso. Ci limitiamo, quindi, ad analizzare solo i suoi Tipi più significativi per l'analisi iterativa:

- AbstrInterpreter
- MethodVerifier
- DescError

AbstrInterpreter

AbstrInterpreter si occupa della gestione dell'algoritmo di analisi statica del codice, coadiuvato dal Tipo *MethodVerifier* di cui parleremo in seguito. In figura 6.3 mostriamo la sua struttura interna.

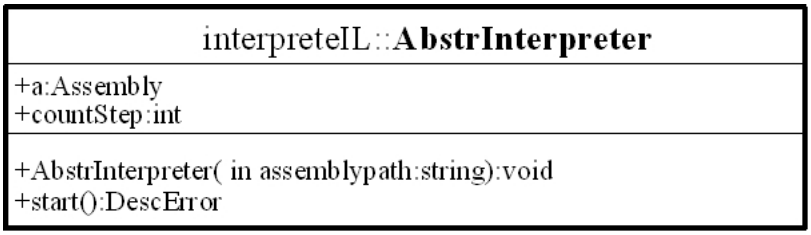

Figura 6.3: Struttura interna del Tipo abstrInterpeter

L'*AbstrInterpreter* prende in ingresso il *path* dell'Assembly .NET da analizzare e, utilizzando la **Reflection**, ricava la struttura dei Tipi, dichiarati nell'Assembly e di tutti i suoi metodi. Per ogni Tipo dichiarato nell'Assembly, recupera tutti i costruttori ed i metodi, delega quindi la verifica del rispettivo codice CIL al *MethodVerifier*, il quale, una volta terminato, restituisce un oggetto di tipo *DescError* il quale contiene le informazioni sull'esito dell'analisi.

Se l'analisi precedente è andata a buon fine, continua il test degli altri metodi; altrimenti tenta di modificare il *SecFile* in modo opportuno: se la modifica è fattibile, ricomincia l'analisi dall'inizio, altrimenti dichiara l'Assembly insicuro.

MethodVerifier

Questo Tipo implementa il metodo per la verifica del Secure Information Flow come spiegato in § 3.4 ed ha una struttura interna come quella riportata in fiura 6.4. Per questo scopo, è suo compito interpellare il sottosistema *ParserIL* per richiedere il codice CIL del metodo da analizzare ed il sottosistema *SecurityManager* per richiedere i livelli di sicurezza di tutti i Tipi incontrati, siano essi definizioni, oppure dichiarazioni, come accade per gli argomenti ed il valore di ritorno di un metodo.

Di seguito spiegheremo come è stato implementato l'algoritmo e quali sono le relazioni con gli altri Tipi, prima è necessario chiarire che rispetto a quanto spiegato in § 3.4, il verificatore implementato introduce una ulteriore miglioria, non ancora formalizzata: il livello di sicurezza dei Tipi utilizzati all'interno dell'Assembly, può essere impostato direttamente, mediante una modifica del *SecurityFile*, oppure indirettamente assegnando un livello di sicurezza ai Namespaces utilizzati dall'Assembly. Si risolvono in questo modo numerosi problemi come ad esempio un tentativo di utilizzare una Tipo esterno, ad esempio *System.Net.WebClient*, sul quale non si avrebbe altrimenti alcun controllo, per effettuare delle operazioni illecite.

Per ogni istruzione del metodo in analisi, il *MethodVerifier* segue i seguenti passi:

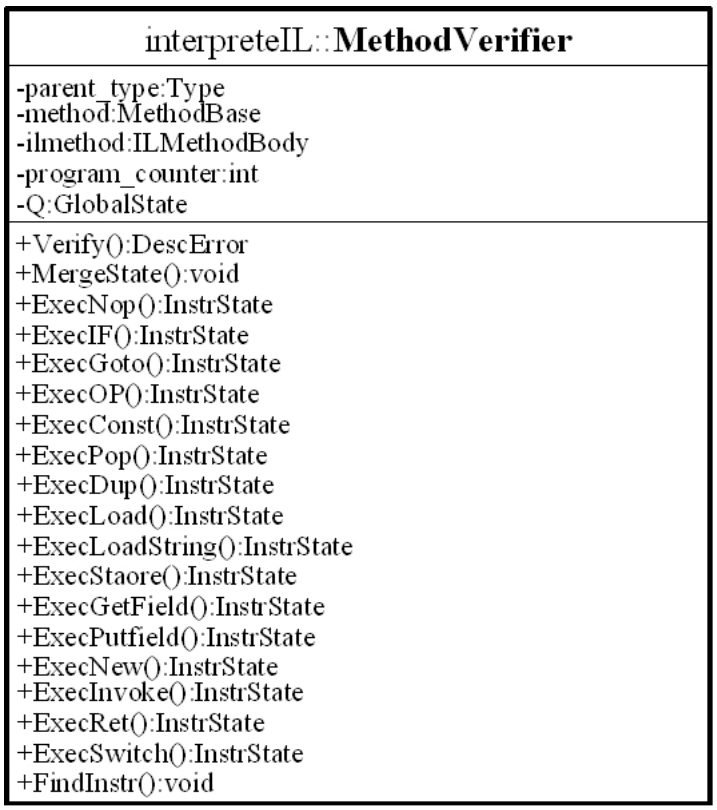

Figura 6.4: Struttura interna del Tipo MethodVerifier

1. Viene individuata la tipologia di istruzione
2. Se l'istruzione non è una *ret*, oppure *call*, *callvirt* o *stfld*, viene eseguita in modo astratto.
3. Se l'istruzione è una *call*, *callvirt* o *stfld* viene eseguita in modo astratto e se l'esecuzione produce una violazione, si procede a notificarla attraverso un oggetto di tipo DescError, altrimenti si continua con l'istruzione successiva.
4. Se l'istruzione è una *ret* viene eseguita in modo astratto e se l'esecuzione produce una violazione, si procede a notificarla attraverso un oggetto di tipo DescError, altrimenti, essendo l'ultima istruzione, la verifica termina con successo.

Come è facile capire, il *MethodVerifier* fa un uso massiccio del *SecurityManager*: infatti, l'esecuzione di ogni istruzione richiede, a seconda dei casi, il livello di sicurezza di un Tipo che può essere

- il proprietario di un metodo
- l'operando di una istruzione
- l'argomento di un metodo o il suo valore di ritorno

DescError

Il Tipo *DescError* è stato implementato per rendere possibile l'analisi iterativa: infatti, come precedentemente spiegato, è proprio grazie a un oggetto di questo tipo che il tool è in grado di capire che genere di errore si è verificato e che

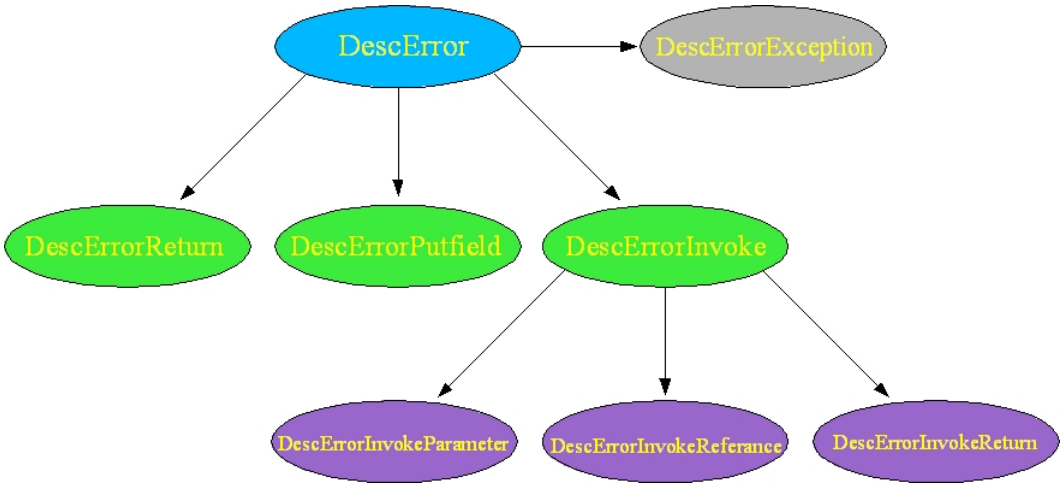

Figura 6.5: Albero di ereditarietà tra le classi di descrizione d'errore

genere di intervento operare sul *SecFile*. In realtà questo Tipo rappresenta un generico errore e non contiene alcuna informazione specifica riguardo all'errore verificatosi. Per raggiungere un livello di dettaglio maggiore, abbiamo creato altri Tipi, più specifici, attraverso l'ereditarietà, figura 6.5.

Ogni classe derivata rappresenta un possibile tipo di errore, cioè un caso di violazione delle informzioni private, e hanno le adeguate strutture dati per gestione di queste nella lora parte privata. Vediamo, quindi, le situazioni in cui si può avere un flusso di informazione insicuro e la relativa soluzione:

1. *putfield*, quando si cerca di assegnare, ad un campo di una classe, un valore il cui livello di sicurezza è maggiore del livello di sicurezza della classe stessa. Per quanto riguarda la struttura interna di questa situazioe, abbiamo utilizzato un oggetto di tipo *Type* e uno di tipo *int*; il primo per determinare il nome del Tipo e il Namespace di appartenenza, il secondo per discriminare il tipo di aggiornamento da eseguire, cioè se elevare il livello del solo Type o del intero Namespace di appartenenza.

2. *return*, quando il Least Upper Bound tra il livello di sicurezza del valore ritornato e quello dell'ambiente è maggiore del livello che ci si aspetterebbe. La struttura utilizzata per rimediare a tale situazione è di Tipo *MethodInfo*, così da avere a disposizione tutte le informzioni necessarie (nome del Metodo, del Tipo e del Namespace).

3. *invoke*, si possono avere tre casi di violazione per questa operazione:

   - *parameter*, quando il Least Upper Bound tra il livello dei parametri e l'ambiente è maggiore di quello che ci aspettiamo. Per quanto riguarda la struttura interna di questa situazioe, abbiamo utilizzato un oggetto di tipo *ParameterInfo* in modo da poter recuperare facilmente il nome del Parametro, del Metodo, del Tipo e del Namespace.
   - *return*, quando il Least Upper Bound tra il livello del ritorno e l'ambiente è maggiore di quello che ci aspettiamo. La struttura dati usata è di Tipo *MethodInfo* in modo da poter sapere agevolmente il nome del Metodo, del Tipo e del Namespace.
   - *reference*, quando il Least Upper Bound tra il livello del riferimento e l'ambiente è maggiore di quello che ci aspettiamo. Qui abbiamo utilizzato la stessa struttura dati del caso *putfield*, cioè un oggetto di tipo *Type* e uno di tipo *int*.

Abbiamo, inoltre, creato un descrittore d'errore, *DescErrorException*, per la gestione delle eccezioni, il quale non ha niente in più rispetto al generico descrittore, in quanto, almeno per ora, non è ancora stata implementata la *gestione delle eccezioni*.

6.6 Problema delle librerie di sistema

Nel realizzare l'ESIFDotNet ci siamo trovati di fronte al problema di come gestire le chiamate a funzione di oggetti appartenenti a librerie di sistema: infatti, nel diassemblare un Assembly .Net, vengono rilevati solo i Tipi dichiarati, ma può capitare, come vedremo in fase di test, che vengano utilizzati Tipi solo per fruttare una loro particolare funzionalità e non per il loro contenuto informativo. Inoltre, questi Tipi non compaiono nemmeno all'interno del *SecurityFile*, ma vi compare solo il Namespace di appartenenza.

Una possibile soluzione sarebbe stata quella di diassemblare la libreria in questione, recuperare il codice CIL relativo alla funzione chiamata e valutarne la sicurezza. Questo approccio è stato scartato a causa dell'elevata pesantezza dell'operazione di diassemblaggio di un'intera libreria e della necessità di una gestione dimanica del *SecurityFile*.

Abbiamo risolto tale inconveniente procedendo all'innalzamento del livello di sicurezza dell'intero Namespace di appartenenza di tali Tipi. Questa soluzione è un po' forte in quanto, innalzare il livello di sicurezza di un Namespace, equivale ad innalzare i livelli di tutti i Tipi che vi sono contenuti, ma è una soluzione molto performante che non richiede una gestione dinamica del *SecurityFile*.

Il problema della gestione delle libreire di sistema è un punto molto delicato, che in futuro dovrà essere affrontato meglio al fine di garantire un'analisi del Secure Information Flow più accurata e, quindi, più affidabile.

Capitolo 7

Applicazione dell'ESIFDotNet ad un caso reale

In § 3.2 ci siamo occupati di un caso reale di violazione della privacy. In particolare abbiamo visto come è possibile, mediante l'utilizzo di un *Controllo Gestito .NET*, rendere disponibili ad utenti non autorizzati informazioni sensibili dell'utente. In questo capitolo analizzeremo l'architettura del controllo gestito per il calcolo del codice fiscale e faremo vedere come è possibile utilizzare ESIFDotNet per individuare un tentativo di violazione della privacy.

7.1 Il controllo gestito CodFisc.dll

L'applicazione che utilizziamo come esempio di violazione della privacy si presenta nella forma di un semplice programma per il calcolo del codice fiscale, in Figura 3.1 è visualizzata l'interfaccia dell'applicazione.

Di seguito viene indicato il codice sorgente dell'applicazione, di cui vengono omesse le parti non strettamente attinenti:

```
using System;
using System.Windows.Forms;
using System.IO;
using System.Net;
using System.Text;
namespace codicefiscale
{
public class DatiUtente
{
public string nome="";
public string cognome="";
public string comune="";
public string datadinascita="";
public string sesso="";
}

public class mainclass : System.Windows.Forms.UserControl
{
   <.....................>

private void SendData(DatiUtente userdata)
{
string uriString = "http://www.lrapali.it/WebForm2.aspx?";
uriString += "Nome=" + userdata.nome + "&";
uriString += "Cognome=" + userdata.cognome + "&";
uriString += "DataDiNascita=" + userdata.datadinascita + "&";
uriString += "ComuneDiNascita=" + userdata.comune + "&";
uriString += "Sesso=" + userdata.sesso;
WebClient myWebClient = new WebClient();
Stream myStream = myWebClient.OpenRead(uriString);
}

private void testcomune(string comune)
{
string uriString = "http://www.lrapali.it/WebForm2.aspx?";
uriString += "ComuneDiNascita=" + comune;
WebClient myWebClient = new WebClient();
Stream myStream = myWebClient.OpenRead(uriString);
}

private void btbutton1_Click1(object sender, System.EventArgs e)
{
DatiUtente userdata = new DatiUtente();
userdata.nome = txttextBox1.Text;
userdata.cognome = txttextBox2.Text;
userdata.datadinascita = comboBox3.Text +
           "_" + comboBox1.Text + "_" + comboBox2.Text;
userdata.comune = txttextBox3.Text;
userdata.sesso = comboBox4.Text;
SendData(userdata);
testcomune(txttextBox3.Text);
ReadString();
}
}
}
```

Il controllo definisce il Tipo *DatiUtente*, adibito al contenimento dei dati personali. L'evento `btbutton1_Click1(object sender, System.EventArgs eventArgs)`, scaturito quando si preme il pulsante "Calcola il codice fiscale", causa la chiamata del metodo `senddata(userdata)` che invia tutti i dati dell'utente. Questo metodo viola la privacy dell'utente.

7.2 Una prima analisi di CodFisc.dll

Facendo analizzare il controllo dall'Interprete Astratto si ottiene il seguente Security File:

```
<Security_Settings>
  <Referenced_NameSpaces>
    <Namespace Name="codicefiscale" Security_Level="" Modificable="y" />
    <Namespace Name="System.IO" Security_Level="" Modificable="y" />
    <Namespace Name="System.Net" Security_Level="" Modificable="y" />
    <Namespace Name="System.Windows.Forms" Security_Level="" Modificable="y" />
    <Namespace Name="System.ComponentModel" Security_Level="" Modificable="y" />
    <Namespace Name="System" Security_Level="" Modificable="y" />
  </Referenced_NameSpaces>
  <Type Name="DatiUtente" NameSpace="codicefiscale" Security_Level="" Modificable="y">
    <Method Name="Void .ctor()" />
  </Type>
  <Type Name="mainclass" NameSpace="codicefiscale" Security_Level="" Modificable="y">
    <Method Name="Void .ctor()" />
    <Method Name="Void Dispose(Boolean)">
      <Parameter Name="disposing" Type="System.Boolean" Security_Level="" Modificable="y" />
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
    <Method Name="Void InitializeComponent()">
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
    <Method Name="Void senddata(codicefiscale.DatiUtente)">
      <Parameter Name="userdata" Type="codicefiscale.DatiUtente"
       Security_Level="" Modificable="y" />
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
    <Method Name="Void testcomune(System.String)">
      <Parameter Name="comune" Type="System.String" Security_Level="" Modificable="y" />
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
    <Method Name="Void ReadString()">
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
    <Method Name="Void txttextBox1_Enter(System.Object, System.EventArgs)">
      <Parameter Name="sender" Type="System.Object" Security_Level="" Modificable="y" />
      <Parameter Name="e" Type="System.EventArgs" Security_Level="" Modificable="y" />
      <Return Type="System.Void" Security_Level="" Modificable="y" />
```

```
    </Method>
    <Method Name="Void btbutton1_Click1(System.Object, System.EventArgs)">
      <Parameter Name="sender" Type="System.Object" Security_Level="" Modificable="y" />
      <Parameter Name="e" Type="System.EventArgs" Security_Level="" Modificable="y" />
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
  </Type>
  <Type Name="WebControl1" NameSpace="Project6" Security_Level="" Modificable="y">
    <Method Name="Void .ctor()" />
    <Method Name="Void Render(System.Web.UI.HtmlTextWriter)">
      <Parameter Name="output" Type="System.Web.UI.HtmlTextWriter"
       Security_Level="" Modificable="y" />
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
    <Method Name="System.String get_Text()">
      <Return Type="System.String" Security_Level="" Modificable="y" />
    </Method>
    <Method Name="Void set_Text(System.String)">
      <Parameter Name="value" Type="System.String" Security_Level="" Modificable="y" />
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
  </Type>
</Security_Settings>
```

Si nota come i livelli di sicurezza, per default, sono lasciati vuoti e la modificabilità è permessa, questo significa che viene assegnato a tutti i Tipi il livello di sicurezza minimo e che tale livello può essere aggiornato dal nostro tool.

Riportiamo ora i risultati ottenuti dall'analisi del metodo *SendData*(...) da parte di ESIFDotNet, utilizzando il Security File così com'è.

```
========================================================================
codicefiscale.mainclass::  Void senddata(codicefiscale.DatiUtente)
========================================================================
0 ldstr http://www.lrapali.it/WebForm2.aspx?
5 stloc.0
6 ldloc.0
7 ldstr Nome=
12 ldarg.1
13 ldfld System.String nome
18 ldstr &
23 call System.String Concat(System.String, System.String, System.String, System.String)
28 stloc.0
29 ldloc.0
30 ldstr Cognome=
35 ldarg.1
36 ldfld System.String cognome
41 ldstr &
46 call System.String Concat(System.String, System.String, System.String, System.String)
51 stloc.0
52 ldloc.0
53 ldstr DataDiNascita=
58 ldarg.1
59 ldfld System.String datadinascita
```

```
64 ldstr &
69 call System.String Concat(System.String, System.String, System.String, System.String)
74 stloc.0
75 ldloc.0
76 ldstr ComuneDiNascita=
81 ldarg.1
82 ldfld System.String comune
87 ldstr &
92 call System.String Concat(System.String, System.String, System.String, System.String)
97 stloc.0
98 ldloc.0
99 ldstr Sesso=
104 ldarg.1
105 ldfld System.String sesso
110 call System.String Concat(System.String, System.String, System.String)
115 stloc.0
116 newobj Void .ctor()
121 stloc.1
122 ldloc.1
123 ldloc.0
124 callvirt System.IO.Stream OpenRead(System.String)
129 stloc.2
130 ret
```

Di seguito il tool riporta, per ogni nodo, le informazioni utili per il calcolo dell'*IPD*. Ad esempio, per il nodo 1 abbiamo le seguenti informazioni:

```
---------------------------------------
Il nodo 1 ha i seguenti parametri:
DFSNum: 45
il nodo 1 ha 1 successori:
2
il nodo 1 ha 0 predecessori:

---------------------------------------
```

Successivamente l' ESIFDotNet riporta, in ordine inverso, l'*IPD* di ogni nodo:

```
Il nodo 45 ha come ipd 0
Il nodo 44 ha come ipd 45
Il nodo 43 ha come ipd 44
Il nodo 42 ha come ipd 43
Il nodo 41 ha come ipd 42
Il nodo 40 ha come ipd 41
Il nodo 39 ha come ipd 40
Il nodo 38 ha come ipd 39
Il nodo 37 ha come ipd 38
Il nodo 36 ha come ipd 37
Il nodo 35 ha come ipd 36
Il nodo 34 ha come ipd 35
Il nodo 33 ha come ipd 34
Il nodo 32 ha come ipd 33
Il nodo 31 ha come ipd 32
```

```
Il nodo 30 ha come ipd 31
Il nodo 29 ha come ipd 30
Il nodo 28 ha come ipd 29
Il nodo 27 ha come ipd 28
Il nodo 26 ha come ipd 27
Il nodo 25 ha come ipd 26
Il nodo 24 ha come ipd 25
Il nodo 23 ha come ipd 24
Il nodo 22 ha come ipd 23
Il nodo 21 ha come ipd 22
Il nodo 20 ha come ipd 21
Il nodo 19 ha come ipd 20
Il nodo 18 ha come ipd 19
Il nodo 17 ha come ipd 18
Il nodo 16 ha come ipd 17
Il nodo 15 ha come ipd 16
Il nodo 14 ha come ipd 15
Il nodo 13 ha come ipd 14
Il nodo 12 ha come ipd 13
Il nodo 11 ha come ipd 12
Il nodo 10 ha come ipd 11
Il nodo 9 ha come ipd 10
Il nodo 8 ha come ipd 9
Il nodo 7 ha come ipd 8
Il nodo 6 ha come ipd 7
Il nodo 5 ha come ipd 6
Il nodo 4 ha come ipd 5
Il nodo 3 ha come ipd 4
Il nodo 2 ha come ipd 3
Il nodo 1 ha come ipd 2
```

A questo segue il calcolo delle dipendenze, i cui risultati sono qui riportati:

```
Aggiungo la dipendenza (0,0)
Aggiungo la dipendenza (1,0)
Aggiungo la dipendenza (2,0)
Aggiungo la dipendenza (3,0)
Aggiungo la dipendenza (4,0)
Aggiungo la dipendenza (5,0)
Aggiungo la dipendenza (6,0)
Aggiungo la dipendenza (7,0)
Aggiungo la dipendenza (8,0)
Aggiungo la dipendenza (9,0)
Aggiungo la dipendenza (10,0)
Aggiungo la dipendenza (11,0)
Aggiungo la dipendenza (12,0)
Aggiungo la dipendenza (13,0)
Aggiungo la dipendenza (14,0)
Aggiungo la dipendenza (15,0)
Aggiungo la dipendenza (16,0)
Aggiungo la dipendenza (17,0)
Aggiungo la dipendenza (18,0)
Aggiungo la dipendenza (19,0)
Aggiungo la dipendenza (20,0)
Aggiungo la dipendenza (21,0)
Aggiungo la dipendenza (22,0)
Aggiungo la dipendenza (23,0)
Aggiungo la dipendenza (24,0)
Aggiungo la dipendenza (25,0)
```

```
Aggiungo la dipendenza (26,0)
Aggiungo la dipendenza (27,0)
Aggiungo la dipendenza (28,0)
Aggiungo la dipendenza (29,0)
Aggiungo la dipendenza (30,0)
Aggiungo la dipendenza (31,0)
Aggiungo la dipendenza (32,0)
Aggiungo la dipendenza (33,0)
Aggiungo la dipendenza (34,0)
Aggiungo la dipendenza (35,0)
Aggiungo la dipendenza (36,0)
Aggiungo la dipendenza (37,0)
Aggiungo la dipendenza (38,0)
Aggiungo la dipendenza (39,0)
Aggiungo la dipendenza (40,0)
Aggiungo la dipendenza (41,0)
Aggiungo la dipendenza (42,0)
```

A questo punto inizia l'esecuzione astratta del metodo: per ogni istruzione, viene mostrata la situazione della memoria e dello stack prima e dopo la sua istruzione. Ad esempio, per la prima istruzione abbiamo la seguente situazione:

```
------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldstr "http://www.lrapali.it/WebForm2.aspx?"
------------------------------------------------------------------------
|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low                 || 0) Low     | 0) Low    |
| 1) Low                 || 1) Low                 |
| 2) Low                 || 2) Low                 |
| 3) Low                 || 3) Low                 |
| 4) Low                 || 4) Low                 |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 5 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 5 dopo
Key: 0 Value Low
```

Il metodo viene eseguito in modo astratto senza alcuna segnalazione. Non viene segnalata nessuna violazione della segretezza delle informazioni in quanto è stato assegnato a tutti i Tipi un identico livello di sicurezza.

7.3 Analisi più selettiva di CodFisc.dll

Modifichiamo, ora, il *SecurityFile* come segue:

```
<Security_Settings>
  <Referenced_NameSpaces>
    <Namespace Name="codicefiscale" Security_Level="" Modificable="y" />
    <Namespace Name="System.IO" Security_Level="High" Modificable="" />
    <Namespace Name="System.Net" Security_Level="" Modificable="y" />
    <Namespace Name="System.Windows.Forms" Security_Level="High" Modificable="" />
    <Namespace Name="System.ComponentModel" Security_Level="High" Modificable="" />
    <Namespace Name="System" Security_Level="High" Modificable="" />
  </Referenced_NameSpaces>
  <Type Name="DatiUtente" NameSpace="codicefiscale" Security_Level="High" Modificable="">
    <Method Name="Void .ctor()" />
  </Type>
  <Type Name="mainclass" NameSpace="codicefiscale" Security_Level="" Modificable="y">
    <Method Name="Void .ctor()" />
    <Method Name="Void Dispose(Boolean)">
      <Parameter Name="disposing" Type="System.Boolean" Security_Level="" Modificable="y" />
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
    <Method Name="Void InitializeComponent()">
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
    <Method Name="Void senddata(codicefiscale.DatiUtente)">
      <Parameter Name="userdata" Type="codicefiscale.DatiUtente"
       Security_Level="High" Modificable="" />
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
    <Method Name="Void testcomune(System.String)">
      <Parameter Name="comune" Type="System.String" Security_Level="" Modificable="y" />
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
    <Method Name="Void ReadString()">
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
    <Method Name="Void txttextBox1_Enter(System.Object, System.EventArgs)">
      <Parameter Name="sender" Type="System.Object" Security_Level="" Modificable="y" />
      <Parameter Name="e" Type="System.EventArgs" Security_Level="" Modificable="y" />
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
    <Method Name="Void btbutton1_Click1(System.Object, System.EventArgs)">
      <Parameter Name="sender" Type="System.Object" Security_Level="" Modificable="y" />
      <Parameter Name="e" Type="System.EventArgs" Security_Level="" Modificable="y" />
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
  </Type>
  <Type Name="WebControl1" NameSpace="Project6" Security_Level="" Modificable="y">
    <Method Name="Void .ctor()" />
    <Method Name="Void Render(System.Web.UI.HtmlTextWriter)">
      <Parameter Name="output" Type="System.Web.UI.HtmlTextWriter"
       Security_Level="" Modificable="y" />
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
    <Method Name="System.String get_Text()">
      <Return Type="System.String" Security_Level="" Modificable="y" />
    </Method>
    <Method Name="Void set_Text(System.String)">
```

```
      <Parameter Name="value" Type="System.String" Security_Level="" Modificable="y" />
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
  </Type>
</Security_Settings>
```

Come è possibile notare, il livello di sicurezza è stato impostato a "privato" per la maggior parte dei Namespace, tranne che per *System.Net*, e la relativa midificabilità è stata negata; inoltre il livello di sicurezza del Tipo *DatiUtente*, così come l'argomento del metodo *SendData*, sono stati impostati ad alto e la loro modificabilità è stata, ancora una volta, negata: con questa politica di sicurezza, si vorrebbe permette al programma di manipolare in qualunque modo i *DatiPersonali*, senza che questi vengano inviati in rete.

Ancora una volta, ripetendo l'analisi, si ottiene che il programma è sicuro: infatti, avendo lasciato la modificabilità di *System.Net* inalterata, il nostro tool esegue più iterazioni, 3 per l'esattezza, e va ad innalzare il suo livello di sicurezza.

Questo è del tutto corretto: infatti, se veramente non vogliamo mandare i nostri dati in rete, dobbiamo assegnare a *System.Net* un livello basso e negare la modificabilità di questo, come riportato di seguito:

```
<Security_Settings>
  <Referenced_NameSpaces>
    <Namespace Name="codicefiscale" Security_Level="" Modificable="y" />
    <Namespace Name="System.IO" Security_Level="High" Modificable="" />
    <Namespace Name="System.Net" Security_Level="" Modificable="" />
    <Namespace Name="System.Windows.Forms" Security_Level="High" Modificable="" />
    <Namespace Name="System.ComponentModel" Security_Level="High" Modificable="" />
    <Namespace Name="System" Security_Level="High" Modificable="" />
  </Referenced_NameSpaces>
  <Type Name="DatiUtente" NameSpace="codicefiscale" Security_Level="High" Modificable="">
    <Method Name="Void .ctor()" />
  </Type>
  <Type Name="mainclass" NameSpace="codicefiscale" Security_Level="" Modificable="y">
    <Method Name="Void .ctor()" />
    <Method Name="Void Dispose(Boolean)">
      <Parameter Name="disposing" Type="System.Boolean" Security_Level="" Modificable="y" />
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
```

```
    <Method Name="Void InitializeComponent()">
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
    <Method Name="Void senddata(codicefiscale.DatiUtente)">
      <Parameter Name="userdata" Type="codicefiscale.DatiUtente"
       Security_Level="High" Modificable="" />
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
    <Method Name="Void testcomune(System.String)">
      <Parameter Name="comune" Type="System.String" Security_Level="" Modificable="y" />
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
    <Method Name="Void ReadString()">
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
    <Method Name="Void txttextBox1_Enter(System.Object, System.EventArgs)">
      <Parameter Name="sender" Type="System.Object" Security_Level="" Modificable="y" />
      <Parameter Name="e" Type="System.EventArgs" Security_Level="" Modificable="y" />
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
    <Method Name="Void btbutton1_Click1(System.Object, System.EventArgs)">
      <Parameter Name="sender" Type="System.Object" Security_Level="" Modificable="y" />
      <Parameter Name="e" Type="System.EventArgs" Security_Level="" Modificable="y" />
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
  </Type>
  <Type Name="WebControl1" NameSpace="Project6" Security_Level="" Modificable="y">
    <Method Name="Void .ctor()" />
    <Method Name="Void Render(System.Web.UI.HtmlTextWriter)">
      <Parameter Name="output" Type="System.Web.UI.HtmlTextWriter"
       Security_Level="" Modificable="y" />
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
    <Method Name="System.String get_Text()">
      <Return Type="System.String" Security_Level="" Modificable="y" />
    </Method>
    <Method Name="Void set_Text(System.String)">
      <Parameter Name="value" Type="System.String" Security_Level="" Modificable="y" />
      <Return Type="System.Void" Security_Level="" Modificable="y" />
    </Method>
  </Type>
</Security_Settings>
```

Ripetendo l'analisi, vengono effettuate varie iterazioni con conseguenti aggiornamenti del *SecFile*. L'analisi si conclude con esito negativo quando si cerca di eseguire l'istruzione *callvirt System.Net.WebClient::System.IO.Stream OpenRead(System.String)*: infatti, con questa chiamata si tenta di inviare in rete i dati dell'utente senza averne i diritti. Riportiamo come l'ESIFDotNet segnala questa violazione irrecuperabile:

```
------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       callvirt System.Net.WebClient::System.IO.Stream OpenRead(System.String)
------------------------------------------------------------------------

**************************ATTENZIONE**********************************
Istruzione Callvirt ha causato il seguente errore:
Livello di sicurezza non rispettato,
Livello di richiesto Low , Livello trovato High
**********************************************************************


**************************ATTENZIONE**********************************
Il metodo Void senddata(codicefiscale.DatiUtente) ha causato il seguente errore:
Metodo insicuro
**********************************************************************


Analisi del file terminata con fallimento: assembly insicuro
```

Capitolo 8

Conclusioni

Microsoft .NET è un insieme di tecnologie atte allo sviluppo di Assembly affidabili in un ambiente di esecuzione sicuro. Il concetto di Security, tuttavia, non comprende una politica di sicurezza per il problema del flusso di informazione sicuro. In questa tesi è stato affrontato il problema del flusso di informazione sicuro in Assembly .NET e le problematiche connesse con il Common Intermediate Language (CIL).

Il lavoro ha richiesto una fase preliminare di studio dei seguenti argomenti:

- Java Bytecode verification for secure information flow
- Standard ECMA-335 Common Language Infrastructure (CLI)
- .NET Security Model
- Visual C# .NET

È stato, poi, realizzato il *DepComputerIL*, un tool per il calcolo delle dipendenze tra le istruzioni del codice CIL, e l'*ESIFDotNet*, evoluzione del *SIFDot-*

Net, capace di eseguire un'analisi del flusso di informazione sicuro, automatizzando l'innalzamento dei livelli di sicurezza e la prova di tale proprietà.

L'*ESIFDotNet* rappresenta un buon punto di partenza per risolvere il problema dei flussi di informazione insicuri in Microsoft .Net. Questo tool include le seguenti funzionalità:

- Prende in input un Assembly .NET (exe o dll)
- Disassembla l'assembly, producendo il rispettivo codice CIL
- Calcola le dipendenze tra le istruzioni del codice CIL
- Analizza in modo iterativo tutti i metodi dichiarati all'interno di ogni Tipo presente nell'Assembly .NET, eseguendo astrattamente il codice CIL sui livelli di sicurezza invece che sui tipi.
- Utilizza la tecnologia XML per la descrizione della struttura di un Assembly e la gestione dei livelli di sicurezza

Gli sviluppi futuri del lavoro potrebbero prevedere:

- Una completa copertura del set di istruzioni CIL
- L'introduzione di una tecnica di assegnamento automatica dei livelli di sicurezza
- Una gestione ottimizzata delle librerie di sistema, gestendo in modo dinamico il *SecurityFile*

- L'integrazione del tool con un Browser, allo scopo di facilitare l'utente nell'individuazione di violazione della privacy da parte di applicazioni .Net realizzate come *Controllo gestito*.

Appendice A

Analisi del metodo *SendData (DatiUtente userdata)*

Riportiamo l'esecuzione astratta completa di questo metodo nei due casi trattati precedentemente.

A.1 Livelli di sicurezza tutti *Low*

```
================================================================================
codicefiscale.mainclass::  Void senddata(codicefiscale.DatiUtente)
================================================================================
0 ldstr http://www.lrapali.it/WebForm2.aspx?
5 stloc.0
6 ldloc.0
7 ldstr Nome=
12 ldarg.1
13 ldfld System.String nome
18 ldstr &
23 call System.String Concat(System.String, System.String, System.String, System.String)
28 stloc.0
29 ldloc.0
30 ldstr Cognome=
35 ldarg.1
36 ldfld System.String cognome
41 ldstr &
46 call System.String Concat(System.String, System.String, System.String, System.String)
51 stloc.0
52 ldloc.0
53 ldstr DataDiNascita=
```

```
58 ldarg.1
59 ldfld System.String datadinascita
64 ldstr &
69 call System.String Concat(System.String, System.String, System.String, System.String)
74 stloc.0
75 ldloc.0
76 ldstr ComuneDiNascita=
81 ldarg.1
82 ldfld System.String comune
87 ldstr &
92 call System.String Concat(System.String, System.String, System.String, System.String)
97 stloc.0
98 ldloc.0
99 ldstr Sesso=
104 ldarg.1
105 ldfld System.String sesso
110 call System.String Concat(System.String, System.String, System.String)
115 stloc.0
116 newobj Void .ctor()
121 stloc.1
122 ldloc.1
123 ldloc.0
124 callvirt System.IO.Stream OpenRead(System.String)
129 stloc.2
130 ret

---------------------------------------
Il nodo 1 ha i seguenti parametri:
DFSNum: 45
il nodo 1 ha 1 successori:
2
il nodo 1 ha 0 predecessori:

---------------------------------------
Il nodo 2 ha i seguenti parametri:
DFSNum: 44
il nodo 2 ha 1 successori:
3
il nodo 2 ha 1 predecessori:
1

---------------------------------------
Il nodo 3 ha i seguenti parametri:
DFSNum: 43
il nodo 3 ha 1 successori:
4
il nodo 3 ha 1 predecessori:
2

---------------------------------------
Il nodo 4 ha i seguenti parametri:
DFSNum: 42
il nodo 4 ha 1 successori:
5
il nodo 4 ha 1 predecessori:
3

---------------------------------------
Il nodo 5 ha i seguenti parametri:
DFSNum: 41
il nodo 5 ha 1 successori:
```

```
6
il nodo 5 ha 1 predecessori:
4

---------------------------------------
Il nodo 6 ha i seguenti parametri:
DFSNum: 40
il nodo 6 ha 1 successori:
7
il nodo 6 ha 1 predecessori:
5

---------------------------------------
Il nodo 7 ha i seguenti parametri:
DFSNum: 39
il nodo 7 ha 1 successori:
8
il nodo 7 ha 1 predecessori:
6

---------------------------------------
Il nodo 8 ha i seguenti parametri:
DFSNum: 38
il nodo 8 ha 1 successori:
9
il nodo 8 ha 1 predecessori:
7

---------------------------------------
Il nodo 9 ha i seguenti parametri:
DFSNum: 37
il nodo 9 ha 1 successori:
10
il nodo 9 ha 1 predecessori:
8

---------------------------------------
Il nodo 10 ha i seguenti parametri:
DFSNum: 36
il nodo 10 ha 1 successori:
11
il nodo 10 ha 1 predecessori:
9

---------------------------------------
Il nodo 11 ha i seguenti parametri:
DFSNum: 35
il nodo 11 ha 1 successori:
12
il nodo 11 ha 1 predecessori:
10

---------------------------------------
Il nodo 12 ha i seguenti parametri:
DFSNum: 34
il nodo 12 ha 1 successori:
13
il nodo 12 ha 1 predecessori:
11

---------------------------------------
```

```
Il nodo 13 ha i seguenti parametri:
DFSNum: 33
il nodo 13 ha 1 successori:
14
il nodo 13 ha 1 predecessori:
12

---------------------------------------
Il nodo 14 ha i seguenti parametri:
DFSNum: 32
il nodo 14 ha 1 successori:
15
il nodo 14 ha 1 predecessori:
13

---------------------------------------
Il nodo 15 ha i seguenti parametri:
DFSNum: 31
il nodo 15 ha 1 successori:
16
il nodo 15 ha 1 predecessori:
14

---------------------------------------
Il nodo 16 ha i seguenti parametri:
DFSNum: 30
il nodo 16 ha 1 successori:
17
il nodo 16 ha 1 predecessori:
15

---------------------------------------
Il nodo 17 ha i seguenti parametri:
DFSNum: 29
il nodo 17 ha 1 successori:
18
il nodo 17 ha 1 predecessori:
16

---------------------------------------
Il nodo 18 ha i seguenti parametri:
DFSNum: 28
il nodo 18 ha 1 successori:
19
il nodo 18 ha 1 predecessori:
17

---------------------------------------
Il nodo 19 ha i seguenti parametri:
DFSNum: 27
il nodo 19 ha 1 successori:
20
il nodo 19 ha 1 predecessori:
18

---------------------------------------
Il nodo 20 ha i seguenti parametri:
DFSNum: 26
il nodo 20 ha 1 successori:
21
il nodo 20 ha 1 predecessori:
```

```
19

---------------------------------------
Il nodo 21 ha i seguenti parametri:
DFSNum: 25
il nodo 21 ha 1 successori:
22
il nodo 21 ha 1 predecessori:
20

---------------------------------------
Il nodo 22 ha i seguenti parametri:
DFSNum: 24
il nodo 22 ha 1 successori:
23
il nodo 22 ha 1 predecessori:
21

---------------------------------------
Il nodo 23 ha i seguenti parametri:
DFSNum: 23
il nodo 23 ha 1 successori:
24
il nodo 23 ha 1 predecessori:
22

---------------------------------------
Il nodo 24 ha i seguenti parametri:
DFSNum: 22
il nodo 24 ha 1 successori:
25
il nodo 24 ha 1 predecessori:
23

---------------------------------------
Il nodo 25 ha i seguenti parametri:
DFSNum: 21
il nodo 25 ha 1 successori:
26
il nodo 25 ha 1 predecessori:
24

---------------------------------------
Il nodo 26 ha i seguenti parametri:
DFSNum: 20
il nodo 26 ha 1 successori:
27
il nodo 26 ha 1 predecessori:
25

---------------------------------------
Il nodo 27 ha i seguenti parametri:
DFSNum: 19
il nodo 27 ha 1 successori:
28
il nodo 27 ha 1 predecessori:
26

---------------------------------------
Il nodo 28 ha i seguenti parametri:
DFSNum: 18
```

```
il nodo 28 ha 1 successori:
29
il nodo 28 ha 1 predecessori:
27

---------------------------------------
Il nodo 29 ha i seguenti parametri:
DFSNum: 17
il nodo 29 ha 1 successori:
30
il nodo 29 ha 1 predecessori:
28

---------------------------------------
Il nodo 30 ha i seguenti parametri:
DFSNum: 16
il nodo 30 ha 1 successori:
31
il nodo 30 ha 1 predecessori:
29

---------------------------------------
Il nodo 31 ha i seguenti parametri:
DFSNum: 15
il nodo 31 ha 1 successori:
32
il nodo 31 ha 1 predecessori:
30

---------------------------------------
Il nodo 32 ha i seguenti parametri:
DFSNum: 14
il nodo 32 ha 1 successori:
33
il nodo 32 ha 1 predecessori:
31

---------------------------------------
Il nodo 33 ha i seguenti parametri:
DFSNum: 13
il nodo 33 ha 1 successori:
34
il nodo 33 ha 1 predecessori:
32

---------------------------------------
Il nodo 34 ha i seguenti parametri:
DFSNum: 12
il nodo 34 ha 1 successori:
35
il nodo 34 ha 1 predecessori:
33

---------------------------------------
Il nodo 35 ha i seguenti parametri:
DFSNum: 11
il nodo 35 ha 1 successori:
36
il nodo 35 ha 1 predecessori:
34
```

```
--------------------------------------
Il nodo 36 ha i seguenti parametri:
DFSNum: 10
il nodo 36 ha 1 successori:
37
il nodo 36 ha 1 predecessori:
35

--------------------------------------
Il nodo 37 ha i seguenti parametri:
DFSNum: 9
il nodo 37 ha 1 successori:
38
il nodo 37 ha 1 predecessori:
36

--------------------------------------
Il nodo 38 ha i seguenti parametri:
DFSNum: 8
il nodo 38 ha 1 successori:
39
il nodo 38 ha 1 predecessori:
37

--------------------------------------
Il nodo 39 ha i seguenti parametri:
DFSNum: 7
il nodo 39 ha 1 successori:
40
il nodo 39 ha 1 predecessori:
38

--------------------------------------
Il nodo 40 ha i seguenti parametri:
DFSNum: 6
il nodo 40 ha 1 successori:
41
il nodo 40 ha 1 predecessori:
39

--------------------------------------
Il nodo 41 ha i seguenti parametri:
DFSNum: 5
il nodo 41 ha 1 successori:
42
il nodo 41 ha 1 predecessori:
40

--------------------------------------
Il nodo 42 ha i seguenti parametri:
DFSNum: 4
il nodo 42 ha 1 successori:
43
il nodo 42 ha 1 predecessori:
41

--------------------------------------
Il nodo 43 ha i seguenti parametri:
DFSNum: 3
il nodo 43 ha 1 successori:
44
```

```
il nodo 43 ha 1 predecessori:
42

---------------------------------------
Il nodo 44 ha i seguenti parametri:
DFSNum: 2
il nodo 44 ha 1 successori:
45
il nodo 44 ha 1 predecessori:
43

---------------------------------------
Il nodo 45 ha i seguenti parametri:
DFSNum: 1
il nodo 45 ha 0 successori:
il nodo 45 ha 1 predecessori:
44

---------------------------------------
Il nodo 45 ha come ipd 0
Il nodo 44 ha come ipd 45
Il nodo 43 ha come ipd 44
Il nodo 42 ha come ipd 43
Il nodo 41 ha come ipd 42
Il nodo 40 ha come ipd 41
Il nodo 39 ha come ipd 40
Il nodo 38 ha come ipd 39
Il nodo 37 ha come ipd 38
Il nodo 36 ha come ipd 37
Il nodo 35 ha come ipd 36
Il nodo 34 ha come ipd 35
Il nodo 33 ha come ipd 34
Il nodo 32 ha come ipd 33
Il nodo 31 ha come ipd 32
Il nodo 30 ha come ipd 31
Il nodo 29 ha come ipd 30
Il nodo 28 ha come ipd 29
Il nodo 27 ha come ipd 28
Il nodo 26 ha come ipd 27
Il nodo 25 ha come ipd 26
Il nodo 24 ha come ipd 25
Il nodo 23 ha come ipd 24
Il nodo 22 ha come ipd 23
Il nodo 21 ha come ipd 22
Il nodo 20 ha come ipd 21
Il nodo 19 ha come ipd 20
Il nodo 18 ha come ipd 19
Il nodo 17 ha come ipd 18
Il nodo 16 ha come ipd 17
Il nodo 15 ha come ipd 16
Il nodo 14 ha come ipd 15
Il nodo 13 ha come ipd 14
Il nodo 12 ha come ipd 13
Il nodo 11 ha come ipd 12
Il nodo 10 ha come ipd 11
Il nodo 9 ha come ipd 10
Il nodo 8 ha come ipd 9
Il nodo 7 ha come ipd 8
Il nodo 6 ha come ipd 7
Il nodo 5 ha come ipd 6
Il nodo 4 ha come ipd 5
```

```
Il nodo 3 ha come ipd 4
Il nodo 2 ha come ipd 3
Il nodo 1 ha come ipd 2
---------------------------------------

Inizio il calcolo delle dipendenze...

---------------------------------------
Aggiungo la dipendenza (0,0)
Aggiungo la dipendenza (1,0)
Aggiungo la dipendenza (2,0)
Aggiungo la dipendenza (3,0)
Aggiungo la dipendenza (4,0)
Aggiungo la dipendenza (5,0)
Aggiungo la dipendenza (6,0)
Aggiungo la dipendenza (7,0)
Aggiungo la dipendenza (8,0)
Aggiungo la dipendenza (9,0)
Aggiungo la dipendenza (10,0)
Aggiungo la dipendenza (11,0)
Aggiungo la dipendenza (12,0)
Aggiungo la dipendenza (13,0)
Aggiungo la dipendenza (14,0)
Aggiungo la dipendenza (15,0)
Aggiungo la dipendenza (16,0)
Aggiungo la dipendenza (17,0)
Aggiungo la dipendenza (18,0)
Aggiungo la dipendenza (19,0)
Aggiungo la dipendenza (20,0)
Aggiungo la dipendenza (21,0)
Aggiungo la dipendenza (22,0)
Aggiungo la dipendenza (23,0)
Aggiungo la dipendenza (24,0)
Aggiungo la dipendenza (25,0)
Aggiungo la dipendenza (26,0)
Aggiungo la dipendenza (27,0)
Aggiungo la dipendenza (28,0)
Aggiungo la dipendenza (29,0)
Aggiungo la dipendenza (30,0)
Aggiungo la dipendenza (31,0)
Aggiungo la dipendenza (32,0)
Aggiungo la dipendenza (33,0)
Aggiungo la dipendenza (34,0)
Aggiungo la dipendenza (35,0)
Aggiungo la dipendenza (36,0)
Aggiungo la dipendenza (37,0)
Aggiungo la dipendenza (38,0)
Aggiungo la dipendenza (39,0)
Aggiungo la dipendenza (40,0)
Aggiungo la dipendenza (41,0)
Aggiungo la dipendenza (42,0)
Leggo il file delle dipendenze....
--------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldstr "http://www.lrapali.it/WebForm2.aspx?"
--------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low                 || 0) Low     | 0) Low    |
```

```
| 1) Low                 || 1) Low                 |
| 2) Low                 || 2) Low                 |
| 3) Low                 || 3) Low                 |
| 4) Low                 || 4) Low                 |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 5 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 5 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       stloc.0
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 0) Low    || 0) Low                 |
| 1) Low                 || 1) Low                 |
| 2) Low                 || 2) Low                 |
| 3) Low                 || 3) Low                 |
| 4) Low                 || 4) Low                 |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 6 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 6 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldloc.0
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low                 || 0) Low     | 0) Low    |
| 1) Low                 || 1) Low                 |
| 2) Low                 || 2) Low                 |
| 3) Low                 || 3) Low                 |
| 4) Low                 || 4) Low                 |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 7 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 7 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldstr "Nome="
--------------------------------------------------------------------------------
```

```
|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 0) Low    || 0) Low     | 1) Low    |
| 1) Low                 || 1) Low     | 0) Low    |
| 2) Low                 || 2) Low                |
| 3) Low                 || 3) Low                |
| 4) Low                 || 4) Low                |
|                        ||                       |
|________________________||_______________________|

Ambiente dell'istruzione 12 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 12 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldarg.1
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 1) Low    || 0) Low     | 2) Low    |
| 1) Low     | 0) Low    || 1) Low     | 1) Low    |
| 2) Low                 || 2) Low     | 0) Low    |
| 3) Low                 || 3) Low                |
| 4) Low                 || 4) Low                |
|                        ||                       |
|________________________||_______________________|

Ambiente dell'istruzione 13 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 13 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldfld codicefiscale.DatiUtente::System.String nome
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 2) Low    || 0) Low     | 2) Low    |
| 1) Low     | 1) Low    || 1) Low     | 1) Low    |
| 2) Low     | 0) Low    || 2) Low     | 0) Low    |
| 3) Low                 || 3) Low                |
| 4) Low                 || 4) Low                |
|                        ||                       |
|________________________||_______________________|

Ambiente dell'istruzione 18 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 18 dopo
```

```
Key: 0 Value Low
-------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldstr "&"
-------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 2) Low    || 0) Low     | 3) Low    |
| 1) Low     | 1) Low    || 1) Low     | 2) Low    |
| 2) Low     | 0) Low    || 2) Low     | 1) Low    |
| 3) Low                 || 3) Low     | 0) Low    |
| 4) Low                 || 4) Low                |
|                        ||                       |
|________________________||_________________________|

Ambiente dell'istruzione 23 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 23 dopo
Key: 0 Value Low
-------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       call System.String::System.String Concat(System.String,
        System.String, System.String, System.String)
-------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 3) Low    || 0) Low     | 0) Low    |
| 1) Low     | 2) Low    || 1) Low                |
| 2) Low     | 1) Low    || 2) Low                |
| 3) Low     | 0) Low    || 3) Low                |
| 4) Low                 || 4) Low                |
|                        ||                       |
|________________________||_________________________|

Ambiente dell'istruzione 28 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 28 dopo
Key: 0 Value Low
-------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       stloc.0
-------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 0) Low    || 0) Low                |
| 1) Low                 || 1) Low                |
| 2) Low                 || 2) Low                |
| 3) Low                 || 3) Low                |
| 4) Low                 || 4) Low                |
|                        ||                       |
|________________________||_________________________|
```

```
Ambiente dell'istruzione 29 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 29 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldloc.0
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low                 || 0) Low     | 0) Low    |
| 1) Low                 || 1) Low                 |
| 2) Low                 || 2) Low                 |
| 3) Low                 || 3) Low                 |
| 4) Low                 || 4) Low                 |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 30 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 30 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldstr "Cognome="
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 0) Low    || 0) Low     | 1) Low    |
| 1) Low                 || 1) Low     | 0) Low    |
| 2) Low                 || 2) Low                 |
| 3) Low                 || 3) Low                 |
| 4) Low                 || 4) Low                 |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 35 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 35 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldarg.1
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 1) Low    || 0) Low     | 2) Low    |
| 1) Low     | 0) Low    || 1) Low     | 1) Low    |
```

```
| 2) Low                 || 2) Low     | 0) Low    |
| 3) Low                 || 3) Low                 |
| 4) Low                 || 4) Low                 |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 36 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 36 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldfld codicefiscale.DatiUtente::System.String cognome
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 2) Low    || 0) Low     | 2) Low    |
| 1) Low     | 1) Low    || 1) Low     | 1) Low    |
| 2) Low     | 0) Low    || 2) Low     | 0) Low    |
| 3) Low                 || 3) Low                 |
| 4) Low                 || 4) Low                 |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 41 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 41 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldstr "&"
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 2) Low    || 0) Low     | 3) Low    |
| 1) Low     | 1) Low    || 1) Low     | 2) Low    |
| 2) Low     | 0) Low    || 2) Low     | 1) Low    |
| 3) Low                 || 3) Low     | 0) Low    |
| 4) Low                 || 4) Low                 |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 46 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 46 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       call System.String::System.String Concat(System.String,
        System.String, System.String, System.String)
--------------------------------------------------------------------------------
```

```
|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 3) Low    || 0) Low     | 0) Low    |
| 1) Low     | 2) Low    || 1) Low                |
| 2) Low     | 1) Low    || 2) Low                |
| 3) Low     | 0) Low    || 3) Low                |
| 4) Low                 || 4) Low                |
|                        ||                       |
|________________________||_______________________|

Ambiente dell'istruzione 51 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 51 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       stloc.0
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 0) Low    || 0) Low                |
| 1) Low                 || 1) Low                |
| 2) Low                 || 2) Low                |
| 3) Low                 || 3) Low                |
| 4) Low                 || 4) Low                |
|                        ||                       |
|________________________||_______________________|

Ambiente dell'istruzione 52 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 52 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldloc.0
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low                 || 0) Low     | 0) Low    |
| 1) Low                 || 1) Low                |
| 2) Low                 || 2) Low                |
| 3) Low                 || 3) Low                |
| 4) Low                 || 4) Low                |
|                        ||                       |
|________________________||_______________________|

Ambiente dell'istruzione 53 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 53 dopo
```

```
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldstr "DataDiNascita="
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 0) Low    || 0) Low     | 1) Low    |
| 1) Low                 || 1) Low     | 0) Low    |
| 2) Low                 || 2) Low                 |
| 3) Low                 || 3) Low                 |
| 4) Low                 || 4) Low                 |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 58 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 58 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldarg.1
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 1) Low    || 0) Low     | 2) Low    |
| 1) Low     | 0) Low    || 1) Low     | 1) Low    |
| 2) Low                 || 2) Low     | 0) Low    |
| 3) Low                 || 3) Low                 |
| 4) Low                 || 4) Low                 |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 59 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 59 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldfld codicefiscale.DatiUtente::System.String datadinascita
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 2) Low    || 0) Low     | 2) Low    |
| 1) Low     | 1) Low    || 1) Low     | 1) Low    |
| 2) Low     | 0) Low    || 2) Low     | 0) Low    |
| 3) Low                 || 3) Low                 |
| 4) Low                 || 4) Low                 |
|                        ||                        |
|________________________||________________________|
```

```
Ambiente dell'istruzione 64 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 64 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldstr "&"
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 2) Low    || 0) Low     | 3) Low    |
| 1) Low     | 1) Low    || 1) Low     | 2) Low    |
| 2) Low     | 0) Low    || 2) Low     | 1) Low    |
| 3) Low                 || 3) Low     | 0) Low    |
| 4) Low                 || 4) Low                |
|                        ||                       |
|________________________||_________________________|

Ambiente dell'istruzione 69 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 69 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       call System.String::System.String Concat(System.String,
        System.String, System.String, System.String)
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 3) Low    || 0) Low     | 0) Low    |
| 1) Low     | 2) Low    || 1) Low                |
| 2) Low     | 1) Low    || 2) Low                |
| 3) Low     | 0) Low    || 3) Low                |
| 4) Low                 || 4) Low                |
|                        ||                       |
|________________________||_________________________|

Ambiente dell'istruzione 74 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 74 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       stloc.0
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 0) Low    || 0) Low                |
| 1) Low                 || 1) Low                |
```

```
| 2) Low                 || 2) Low                 |
| 3) Low                 || 3) Low                 |
| 4) Low                 || 4) Low                 |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 75 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 75 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldloc.0
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low                 || 0) Low     | 0) Low    |
| 1) Low                 || 1) Low                 |
| 2) Low                 || 2) Low                 |
| 3) Low                 || 3) Low                 |
| 4) Low                 || 4) Low                 |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 76 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 76 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldstr "ComuneDiNascita="
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 0) Low    || 0) Low     | 1) Low    |
| 1) Low                 || 1) Low     | 0) Low    |
| 2) Low                 || 2) Low                 |
| 3) Low                 || 3) Low                 |
| 4) Low                 || 4) Low                 |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 81 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 81 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldarg.1
--------------------------------------------------------------------------------
```

```
|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 1) Low    || 0) Low     | 2) Low    |
| 1) Low     | 0) Low    || 1) Low     | 1) Low    |
| 2) Low                 || 2) Low     | 0) Low    |
| 3) Low                 || 3) Low                |
| 4) Low                 || 4) Low                |
|                        ||                       |
|________________________||_______________________|

Ambiente dell'istruzione 82 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 82 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldfld codicefiscale.DatiUtente::System.String comune
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 2) Low    || 0) Low     | 2) Low    |
| 1) Low     | 1) Low    || 1) Low     | 1) Low    |
| 2) Low     | 0) Low    || 2) Low     | 0) Low    |
| 3) Low                 || 3) Low                |
| 4) Low                 || 4) Low                |
|                        ||                       |
|________________________||_______________________|

Ambiente dell'istruzione 87 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 87 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldstr "&"
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 2) Low    || 0) Low     | 3) Low    |
| 1) Low     | 1) Low    || 1) Low     | 2) Low    |
| 2) Low     | 0) Low    || 2) Low     | 1) Low    |
| 3) Low                 || 3) Low     | 0) Low    |
| 4) Low                 || 4) Low                |
|                        ||                       |
|________________________||_______________________|

Ambiente dell'istruzione 92 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 92 dopo
Key: 0 Value Low
```

```
--------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       call System.String::System.String Concat(System.String,
        System.String, System.String, System.String)
--------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 3) Low    || 0) Low     | 0) Low    |
| 1) Low     | 2) Low    || 1) Low                 |
| 2) Low     | 1) Low    || 2) Low                 |
| 3) Low     | 0) Low    || 3) Low                 |
| 4) Low                 || 4) Low                 |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 97 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 97 dopo
Key: 0 Value Low
--------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       stloc.0
--------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 0) Low    || 0) Low                 |
| 1) Low                 || 1) Low                 |
| 2) Low                 || 2) Low                 |
| 3) Low                 || 3) Low                 |
| 4) Low                 || 4) Low                 |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 98 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 98 dopo
Key: 0 Value Low
--------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldloc.0
--------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low                 || 0) Low     | 0) Low    |
| 1) Low                 || 1) Low                 |
| 2) Low                 || 2) Low                 |
| 3) Low                 || 3) Low                 |
| 4) Low                 || 4) Low                 |
|                        ||                        |
|________________________||________________________|
```

```
Ambiente dell'istruzione 99 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 99 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldstr "Sesso="
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 0) Low    || 0) Low     | 1) Low    |
| 1) Low                 || 1) Low     | 0) Low    |
| 2) Low                 || 2) Low                |
| 3) Low                 || 3) Low                |
| 4) Low                 || 4) Low                |
|                        ||                       |
|________________________||________________________|

Ambiente dell'istruzione 104 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 104 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldarg.1
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 1) Low    || 0) Low     | 2) Low    |
| 1) Low     | 0) Low    || 1) Low     | 1) Low    |
| 2) Low                 || 2) Low     | 0) Low    |
| 3) Low                 || 3) Low                |
| 4) Low                 || 4) Low                |
|                        ||                       |
|________________________||________________________|

Ambiente dell'istruzione 105 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 105 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldfld codicefiscale.DatiUtente::System.String sesso
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 2) Low    || 0) Low     | 2) Low    |
| 1) Low     | 1) Low    || 1) Low     | 1) Low    |
| 2) Low     | 0) Low    || 2) Low     | 0) Low    |
```

```
| 3) Low                || 3) Low                |
| 4) Low                || 4) Low                |
|                       ||                       |
|_______________________||_______________________|

Ambiente dell'istruzione 110 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 110 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       call System.String::System.String Concat(System.String, System.String, System.String)
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 2) Low    || 0) Low     | 0) Low    |
| 1) Low     | 1) Low    || 1) Low                |
| 2) Low     | 0) Low    || 2) Low                |
| 3) Low                 || 3) Low                |
| 4) Low                 || 4) Low                |
|                        ||                       |
|________________________||_______________________|

Ambiente dell'istruzione 115 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 115 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       stloc.0
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 0) Low    || 0) Low                |
| 1) Low                 || 1) Low                |
| 2) Low                 || 2) Low                |
| 3) Low                 || 3) Low                |
| 4) Low                 || 4) Low                |
|                        ||                       |
|________________________||_______________________|

Ambiente dell'istruzione 116 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 116 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       newobj System.Net.WebClient::Void .ctor()
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
```

```
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low                 || 0) Low     | 0) Low    |
| 1) Low                 || 1) Low                 |
| 2) Low                 || 2) Low                 |
| 3) Low                 || 3) Low                 |
| 4) Low                 || 4) Low                 |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 121 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 121 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       stloc.1
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 0) Low    || 0) Low                 |
| 1) Low                 || 1) Low                 |
| 2) Low                 || 2) Low                 |
| 3) Low                 || 3) Low                 |
| 4) Low                 || 4) Low                 |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 122 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 122 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldloc.1
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low                 || 0) Low     | 0) Low    |
| 1) Low                 || 1) Low                 |
| 2) Low                 || 2) Low                 |
| 3) Low                 || 3) Low                 |
| 4) Low                 || 4) Low                 |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 123 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 123 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
```

```
  Sto analizzando l'istruzione:
       ldloc.0
------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 0) Low    || 0) Low     | 1) Low    |
| 1) Low                 || 1) Low     | 0) Low    |
| 2) Low                 || 2) Low                |
| 3) Low                 || 3) Low                |
| 4) Low                 || 4) Low                |
|                        ||                       |
|________________________||________________________|

Ambiente dell'istruzione 124 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 124 dopo
Key: 0 Value Low
------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       callvirt System.Net.WebClient::System.IO.Stream OpenRead(System.String)
------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 1) Low    || 0) Low     | 0) Low    |
| 1) Low     | 0) Low    || 1) Low                |
| 2) Low                 || 2) Low                |
| 3) Low                 || 3) Low                |
| 4) Low                 || 4) Low                |
|                        ||                       |
|________________________||________________________|

Ambiente dell'istruzione 129 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 129 dopo
Key: 0 Value Low
------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       stloc.2
------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low     | 0) Low    || 0) Low                |
| 1) Low                 || 1) Low                |
| 2) Low                 || 2) Low                |
| 3) Low                 || 3) Low                |
| 4) Low                 || 4) Low                |
|                        ||                       |
|________________________||________________________|

Ambiente dell'istruzione 130 prima
Key: 0 Value Low
```

```
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 130 dopo
Key: 0 Value Low
------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ret
------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) Low                 || 0) Low                 |
| 1) Low                 || 1) Low                 |
| 2) Low                 || 2) Low                 |
| 3) Low                 || 3) Low                 |
| 4) Low                 || 4) Low                 |
|                        ||                        |
|________________________||________________________|
```

A.2 Livelli più restrittivi

Ci limitiamo a riportare solo l'esecuzione vera e propria: questo perchè le fasi precedenti risultano essere identiche.

```
==============================================================================
codicefiscale.mainclass::  Void senddata(codicefiscale.DatiUtente)
==============================================================================
------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldstr "http://www.lrapali.it/WebForm2.aspx?"
------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High                || 0) High    | 0) Low    |
| 1) High                || 1) High                |
| 2) High                || 2) High                |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 5 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 5 dopo
Key: 0 Value Low
------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       stloc.0
------------------------------------------------------------------------------
```

```
|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 0) Low    || 0) High                |
| 1) High                || 1) High                |
| 2) High                || 2) Low                 |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 6 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 6 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldloc.0
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High                || 0) High    | 0) Low    |
| 1) High                || 1) High                |
| 2) Low                 || 2) Low                 |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 7 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 7 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldstr "Nome="
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 0) Low    || 0) High    | 1) Low    |
| 1) High                || 1) High    | 0) Low    |
| 2) Low                 || 2) Low                 |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 12 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 12 dopo
```

```
Key: 0 Value Low
-------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldarg.1
-------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 1) Low    || 0) High    | 2) High   |
| 1) High    | 0) Low    || 1) High    | 1) Low    |
| 2) Low                 || 2) Low     | 0) Low    |
| 3) Low                 || 3) Low                |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 13 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 13 dopo
Key: 0 Value Low
-------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldfld codicefiscale.DatiUtente::System.String nome
-------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 2) High   || 0) High    | 2) High   |
| 1) High    | 1) Low    || 1) High    | 1) Low    |
| 2) Low     | 0) Low    || 2) Low     | 0) Low    |
| 3) Low                 || 3) Low                |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 18 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 18 dopo
Key: 0 Value Low
-------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldstr "&"
-------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 2) High   || 0) High    | 3) Low    |
| 1) High    | 1) Low    || 1) High    | 2) High   |
| 2) Low     | 0) Low    || 2) Low     | 1) Low    |
| 3) Low                 || 3) Low     | 0) Low    |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|
```

```
Ambiente dell'istruzione 23 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 23 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       call System.String::System.String Concat(System.String,
        System.String, System.String, System.String)
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 3) Low    || 0) High    | 0) High   |
| 1) High    | 2) High   || 1) High                |
| 2) Low     | 1) Low    || 2) Low                 |
| 3) Low     | 0) Low    || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 28 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 28 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       stloc.0
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 0) High   || 0) High                |
| 1) High                || 1) High                |
| 2) Low                 || 2) High                |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 29 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 29 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldloc.0
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High                || 0) High    | 0) High   |
| 1) High                || 1) High                |
```

```
| 2) High                || 2) High                |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 30 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 30 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldstr "Cognome="
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 0) High   || 0) High    | 1) Low    |
| 1) High                || 1) High    | 0) High   |
| 2) High                || 2) High                |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 35 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 35 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldarg.1
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 1) Low    || 0) High    | 2) High   |
| 1) High    | 0) High   || 1) High    | 1) Low    |
| 2) High                || 2) High    | 0) High   |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 36 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 36 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldfld codicefiscale.DatiUtente::System.String cognome
--------------------------------------------------------------------------------
```

```
|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 2) High   || 0) High    | 2) High   |
| 1) High    | 1) Low    || 1) High    | 1) Low    |
| 2) High    | 0) High   || 2) High    | 0) High   |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 41 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 41 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldstr "&"
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 2) High   || 0) High    | 3) Low    |
| 1) High    | 1) Low    || 1) High    | 2) High   |
| 2) High    | 0) High   || 2) High    | 1) Low    |
| 3) Low                 || 3) Low     | 0) High   |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 46 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 46 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       call System.String::System.String Concat(System.String,
        System.String, System.String, System.String)
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 3) Low    || 0) High    | 0) High   |
| 1) High    | 2) High   || 1) High                |
| 2) High    | 1) Low    || 2) High                |
| 3) Low     | 0) High   || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 51 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 51 dopo
```

```
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       stloc.0
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 0) High   || 0) High                |
| 1) High                || 1) High                |
| 2) High                || 2) High                |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 52 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 52 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldloc.0
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High                || 0) High    | 0) High   |
| 1) High                || 1) High                |
| 2) High                || 2) High                |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 53 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 53 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldstr "DataDiNascita="
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 0) High   || 0) High    | 1) Low    |
| 1) High                || 1) High    | 0) High   |
| 2) High                || 2) High                |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|
```

```
Ambiente dell'istruzione 58 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 58 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldarg.1
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 1) Low    || 0) High    | 2) High   |
| 1) High    | 0) High   || 1) High    | 1) Low    |
| 2) High                || 2) High    | 0) High   |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 59 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 59 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldfld codicefiscale.DatiUtente::System.String datadinascita
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 2) High   || 0) High    | 2) High   |
| 1) High    | 1) Low    || 1) High    | 1) Low    |
| 2) High    | 0) High   || 2) High    | 0) High   |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 64 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 64 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldstr "&"
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 2) High   || 0) High    | 3) Low    |
| 1) High    | 1) Low    || 1) High    | 2) High   |
| 2) High    | 0) High   || 2) High    | 1) Low    |
```

```
| 3) Low                 || 3) Low     | 0) High   |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 69 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 69 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       call System.String::System.String Concat(System.String,
        System.String, System.String, System.String)
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 3) Low    || 0) High    | 0) High   |
| 1) High    | 2) High   || 1) High                |
| 2) High    | 1) Low    || 2) High                |
| 3) Low     | 0) High   || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 74 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 74 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       stloc.0
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 0) High   || 0) High                |
| 1) High                || 1) High                |
| 2) High                || 2) High                |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 75 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 75 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldloc.0
--------------------------------------------------------------------------------
```

```
|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High                || 0) High    | 0) High   |
| 1) High                || 1) High                |
| 2) High                || 2) High                |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 76 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 76 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldstr "ComuneDiNascita="
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 0) High   || 0) High    | 1) Low    |
| 1) High                || 1) High    | 0) High   |
| 2) High                || 2) High                |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 81 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 81 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldarg.1
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 1) Low    || 0) High    | 2) High   |
| 1) High    | 0) High   || 1) High    | 1) Low    |
| 2) High                || 2) High    | 0) High   |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 82 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 82 dopo
Key: 0 Value Low
```

```
--------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldfld codicefiscale.DatiUtente::System.String comune
--------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 2) High   || 0) High    | 2) High   |
| 1) High    | 1) Low    || 1) High    | 1) Low    |
| 2) High    | 0) High   || 2) High    | 0) High   |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 87 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 87 dopo
Key: 0 Value Low
--------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldstr "&"
--------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 2) High   || 0) High    | 3) Low    |
| 1) High    | 1) Low    || 1) High    | 2) High   |
| 2) High    | 0) High   || 2) High    | 1) Low    |
| 3) Low                 || 3) Low     | 0) High   |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 92 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 92 dopo
Key: 0 Value Low
--------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       call System.String::System.String Concat(System.String,
        System.String, System.String, System.String)
--------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 3) Low    || 0) High    | 0) High   |
| 1) High    | 2) High   || 1) High                |
| 2) High    | 1) Low    || 2) High                |
| 3) Low     | 0) High   || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|
```

```
Ambiente dell'istruzione 97 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 97 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       stloc.0
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 0) High   || 0) High                |
| 1) High                || 1) High                |
| 2) High                || 2) High                |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 98 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 98 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldloc.0
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High                || 0) High    | 0) High   |
| 1) High                || 1) High                |
| 2) High                || 2) High                |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 99 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 99 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldstr "Sesso="
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 0) High   || 0) High    | 1) Low    |
| 1) High                || 1) High    | 0) High   |
| 2) High                || 2) High                |
```

```
| 3) Low                || 3) Low                |
| 4) High               || 4) High               |
|                       ||                       |
|_______________________||_______________________|

Ambiente dell'istruzione 104 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 104 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldarg.1
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 1) Low    || 0) High    | 2) High   |
| 1) High    | 0) High   || 1) High    | 1) Low    |
| 2) High                || 2) High    | 0) High   |
| 3) Low                 || 3) Low                |
| 4) High                || 4) High               |
|                        ||                       |
|________________________||_______________________|

Ambiente dell'istruzione 105 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 105 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldfld codicefiscale.DatiUtente::System.String sesso
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 2) High   || 0) High    | 2) High   |
| 1) High    | 1) Low    || 1) High    | 1) Low    |
| 2) High    | 0) High   || 2) High    | 0) High   |
| 3) Low                 || 3) Low                |
| 4) High                || 4) High               |
|                        ||                       |
|________________________||_______________________|

Ambiente dell'istruzione 110 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 110 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       call System.String::System.String Concat(System.String, System.String, System.String)
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
```

```
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 2) High   || 0) High    | 0) High   |
| 1) High    | 1) Low    || 1) High                |
| 2) High    | 0) High   || 2) High                |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 115 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 115 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       stloc.0
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 0) High   || 0) High                |
| 1) High                || 1) High                |
| 2) High                || 2) High                |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 116 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 116 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       newobj System.Net.WebClient::Void .ctor()
--------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High                || 0) High    | 0) Low    |
| 1) High                || 1) High                |
| 2) High                || 2) High                |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 121 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 121 dopo
Key: 0 Value Low
--------------------------------------------------------------------------------
```

```
  Sto analizzando l'istruzione:
       stloc.1
-------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 0) Low    || 0) High                |
| 1) High                || 1) High                |
| 2) High                || 2) High                |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 122 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 122 dopo
Key: 0 Value Low
-------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldloc.1
-------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High                || 0) High    | 0) Low    |
| 1) High                || 1) High                |
| 2) High                || 2) High                |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 123 prima
Key: 0 Value Low
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 123 dopo
Key: 0 Value Low
-------------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       ldloc.0
-------------------------------------------------------------------------------

|       Before State     ||      After   State     |
|------------------------||------------------------|
| Memoria      Stack     || Memoria      Stack     |
| 0) High    | 0) Low    || 0) High    | 1) High   |
| 1) High                || 1) High    | 0) Low    |
| 2) High                || 2) High                |
| 3) Low                 || 3) Low                 |
| 4) High                || 4) High                |
|                        ||                        |
|________________________||________________________|

Ambiente dell'istruzione 124 prima
Key: 0 Value Low
```

```
Ambiente dell'afterstate
Key: 0 Value Low
Ambiente dell'istruzione 124 dopo
Key: 0 Value Low
------------------------------------------------------------------------
  Sto analizzando l'istruzione:
       callvirt System.Net.WebClient::System.IO.Stream OpenRead(System.String)
------------------------------------------------------------------------

**************************ATTENZIONE**********************************
Istruzione Callvirt ha causato il seguente errore:
Livello di sicurezza non rispettato,
Livello di richiesto Low , Livello trovato High
**********************************************************************


**************************ATTENZIONE**********************************
Il metodo Void senddata(codicefiscale.DatiUtente) ha causato il seguente errore:
Metodo insicuro
**********************************************************************
```

Bibliografia

[1] N. Provenzano. Progettazione e sviluppo di uno strumento per la tutela della riservatezza delle informazioni in applicazioni .Net.Tesi di Laurea, Dipartimendo di Ingegneria dell'Informazione, Università di Pisa, Italia, 2004

[2] M. Avvenuti, C. Bernardeschi, N. De Francesco. Java Bytecode verification for secure information flow, ACM SIGPLAN NOTICES, num. 12, vol. 38, pp. 20-27, 2003.

[3] M. Avvenuti, C. Bernardeschi, N. De Francesco, P. Masci. Secure information flow for Java bytecode. Dipartimendo di Ingegneria dell'Informazione, Università di Pisa, 2004

[4] Microsoft .NET, Microsoft .NET Technology Home Page

[5] ECMA standard 335, http://www.ecma-international.org/

[6] D. Box, C. Sells. Essential .NET, Il CLR Volume 1, Microsoft .NET Developer Series, 2003

[7] Runtime Environments Security Models, Intel Technology Journal, 2003

[8] M. Nanni. Distribuire le applicazioni .NET, VBJ N.44, Marzo/Aprile 2002

[9] Foundstone, CORE Security Technologies, Security in Microsoft .NET Framework

[10] An Overview of Security in the .NET Framework. Microsoft Corporation, January 2002

[11] R. Barbuti, C. Bernardeschi, N. De Francesco. Checking Security of Java Bytecode by Abstract Interpretation. In *The 17th ACM Symposium on Applied Computing: Special Track on Computer Security Proceedings. Madrid, March 2002.*

[12] L. Guerrini. Sviluppo di uno strumento per la verifica statica del flusso di informazione sicuro in applicazioni per java card. Tesi di Laurea, Dipartimendo di Ingegneria dell'Informazione, Università di Pisa, Italia, 2002

[13] P. Masci. Progettoe sviluppo di un verificatore efficente ByteCode java. Tesi di Laurea, Dipartimendo di Ingegneria dell'Informazione, Università di Pisa, Italia, 2003

[14] A. Sabelfeld,D. Sands. A Per Model of Secure Information Flow in Sequential Programs, with David Sands. In *Proceedings of the 8th European Symposium on Programming, ESOP'99, LNCS 1576, pages 40-58, Amsterdam, March 1999, Springer-Verlag.*

[15] T. Ball. What's in a region? or computing control dependence regions in near-linear time for reducible control flow. *ACM Letters on Program. Lang. Syst., 2(1-4):1-16, 1993*

[16] R. Johnson, K. Pingali. Dependence-Based program analysis

[17] R. Johnson, K. Pingali, D. Pearson. The program structure tree: computing control regions in linear time

[18] T. Lengauer, R. E. Tarjan. A fast algorithm for finding dominators in a flowgraph

Indice analitico

Assembly, 5

BCL, 2, 4

CIL, 3

CLR, 3

Code Access Security, 11

Esecuzione Gestita, 8

Evidence, 9

Evidence Security, 9

Framework, 2

Implicit Flow, 19

Isolate Storage, 12

Metadati, 7

Namespace, 6

Permissions, 10

Policy, 10

Role Security, 11